FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Meta-Heuristic Local Planning

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Computational Intelligence

eingereicht von

Markus Suchi
Matrikelnummer 9103533

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.-Prof. Dipl.-Ing. Dr. techn. Markus Vincze
Mitwirkung: Dipl.-Ing. Dr. techn. Markus Bader

Wien, 21.11.2014

(Unterschrift Markus Suchi)

(Unterschrift Betreuung)

Erklärung zur Verfassung der Arbeit

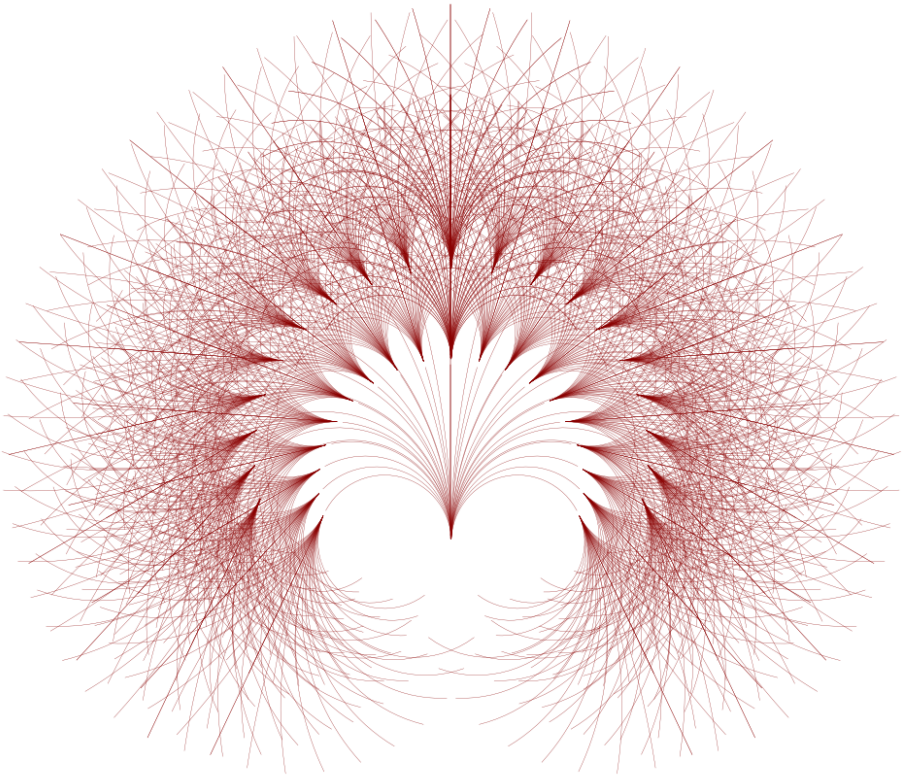
Markus Suchi
Karl Heinz Straße 67/11/51, 1230 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Markus Suchi)

META HEURISTIC LOCAL PLANNING

MARKUS SUCHI



November 2014

ABSTRACT

Navigation is a crucial task for mobile robots driving through dynamic environments. Besides the difficulties involved in finding a way from a starting to some goal location, the problem gets more difficult if unknown objects, dynamics of the robotic vehicle and uncertainties from sensor readings have to be taken into account.

To guarantee a safe passage common approaches divide the navigation task into two parts - global and local planning. A global planner finds an initial path to the desired goal location based on a previous obtained map. The retrieved path is used as a guide for a local path planning component. This local path planner uses so called local information obtained through recent sensor readings and applies obstacle avoidance strategies to safely and efficiently follow the guide as precise as possible.

Very effective local planning methods like the Dynamic Window Approach (DWA) or Trajectory Roll-out are based on sampling the control space of the robot. For a short amount of time the application of these controls is simulated generating corresponding trajectories. By using appropriate cost functions the resulting trajectories are weighted and the best one yields the optimal target values for the motor controller.

The goal of this thesis is to analyze these approaches and improve the performance of local planners by applying well known meta-heuristic search strategies in the trajectory selection process.

For this purpose an introduction to local planning and obstacle avoidance methods is presented, followed by a discussion of applicable single solution based meta-heuristics.

Approaches based on Iterated Local Search (ILS), Variable Neighborhood Search (VNS) and Tabu Search are implemented and tested using a sample planner based on DWA. These algorithms are analyzed and evaluated using random created instances of sensor maps. Results are documenting a significant increase in performance compared to the brute force evaluation commonly used in local planner.

With the gained knowledge of these tests the (VNS) approach is selected to substitute the selection process in a popular implementation of local planner within the Robot Operating System (ROS). Two algorithms based on trajectory Roll-out (VNS-ROL) and DWA (VNS-DWA) are developed and evaluated using a sophisticated simulation engine. The altered planner outperform the original implementations on all tested instances.

ZUSAMMENFASSUNG

Die Fähigkeit in dynamischen Umgebungen zu navigieren ist eine wesentliche Funktion für mobile Roboter. Neben der Schwierigkeit einen Weg von einem Startpunkt zu einem Zielpunkt zu finden, erhöhen sich die Anforderungen an die Planung wenn Hindernisse und Messunsicherheiten vom Roboter während der Navigtion berücksichtigt werden müssen.

Um eine sichere Fahrt zu garantieren, wird die Navigation üblicherweise in zwei Schritte aufgeteilt - globale und lokale Planung. Ein globaler Planer findet anhand einer vorher erstellten Karte einen ersten Weg zum gewünschten Ziel. Der gefundene Weg fungiert dann als Orientierungshilfe für eine lokale Wegplanungskomponente. Der lokale Planer verwendet unter Berücksichtigung der aktuellsten Sensormessungen sogenannte lokale Informationen und Strategien zur Kollisionsvermeidung um sicher und effizient der Orientierungshilfe so exakt wie möglich zu folgen.

Sehr effektive lokale Planungsmethoden wie Dynamic Window Approach (DWA) oder Trajectory Roll-out basieren auf einem Sampling des Controlspace des Roboters. Für einen kurzen Zeitraum wird die Applikation von Steuerwerten simuliert und die resultierenden Trajektorien generiert. Mithilfe einer geeigneten Kostenfunktion werden die Trajektorien gewichtet und die Steuerwerte der besten Trajektorie werden and die Motorsteuerung weitergeleitet.

Das Ziel der vorliegenden Arbeit ist eine Analyse und eine Verbesserung der Leistung von lokalen Planern durch Einsatz bekannter metaheuristischer Suchstrategien bei der Selektion der Trajektorien.

Zu diesem Zweck wird eine Einführung zu lokalen Planungsmethoden und Methoden der Kollisionsvermeidung präsentiert, gefolgt von einer Diskussion zu „single solution based “ Metaheuristiken. Die präsentierten Ansätze basierend auf Iterated Local Search (ILS), Variable Neighborhood Search (VNS) und Tabu Search werden implementiert und mit einem auf DWA basierenden einfachen Planer getestet. Diese Algorithmen werden anhand zufällig erzeugter Sensorkarten analysiert und ausgewertet. Die Resultate dokumentieren eine signifikante Verbesserung der Leistung verglichen mit der Brute-Force-Methode, die üblicherweise bei dieser Art lokaler Planung verwendet wird.

Mit dem durch diese Untersuchungen erworbenen Wissen wird der VNS Ansatz gewählt um den Selektionsprozess in einer existierenden Implementierung von lokalen Planern innerhalb des Robot Operating Systems (ROS) zu ersetzen. Zwei Algorithmen basierend auf Trajectory roll-out (VNS-ROL) und DWA (VNS-DWA) werden entwickelt und mit einer modernen Simulationsoftware evaluiert. Die Leistung der adaptierten Planer übertrifft dabei die ursprünglichen Implementierungen in allen getesteten Szenarien.

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

Part I

INTRODUCTION

1 INTRODUCTION

Navigation and planning are essential for mobile robots to act in out- and indoor environments. From self driving cars navigating 132 miles through the Mojave desert, or mobile robots handling goods in distribution centers and warehouses to mobile robots used for planetary exploration, autonomous robots have conquered nearly every place on earth and beyond.

Articles like [48] and [19] present the relationship between the environmental complexity and the computational on-board processing power needed to tackle the navigation problem. The self driving car Stanley has a six processor computing platform provided by Intel whilst Kiva robots are using low cost DSP's for navigation and vision processing[1] to drive within a known environment.

While the application domains and computational powers varies strongly between the robotic systems, all of them have to move safely and efficient from one location to another.

The method proposed in this thesis improves navigation for mobile robots by combining local planning methods with search strategies based on meta-heuristic algorithms.

This enables the local planner to:

- run at a higher frequency
- simulate trajectories for a longer time interval
- moving the robot at higher speed
- investigate a larger amount of trajectories
- use a higher costmap resolution

In order to give an impression of the importance of navigation and planning in the field of mobile robotics the next section gives some motivation and application examples.

1.1 MOTIVATION AND APPLICATIONS

The goal of navigation encompasses the ability of robots to find a series of actions based on its knowledge of the environment and sensor values to reach

1 Kiva Systems Uses "Smart" Blackfin-powered Robots for Warehouse Navigation | Analog Devices: http://www.analog.com/en/content/kiva_systems_bf548/fca.html

its goal position in an effective and efficient manner. The resulting series of actions is called a *plan*.

To ensure safety and flexibility in the presence of obstacles in a dynamic environment *obstacle avoidance* is used to alter plans during execution and generating of collision free trajectories.

A common strategy to deal with complex navigation problems is to the divide the planning task into a global and a local planning problem [27].

Global path-planning usually operates on a simplified representation of the environment and the robot itself (e.g. static map, circle representation of the robots outline) to efficiently compute an optimal shortest path using variants of Dijkstra's [8] or A* [22] algorithm, ignoring kinematic and acceleration constraints of the robot. In succession the retrieved global path is used by a local planner for guiding the robot through the environment. Figure 1 illustrates the view of the environment from a robot perspective together with a global and local plan, which enables the robot to drive autonomous within a lab/office environment.

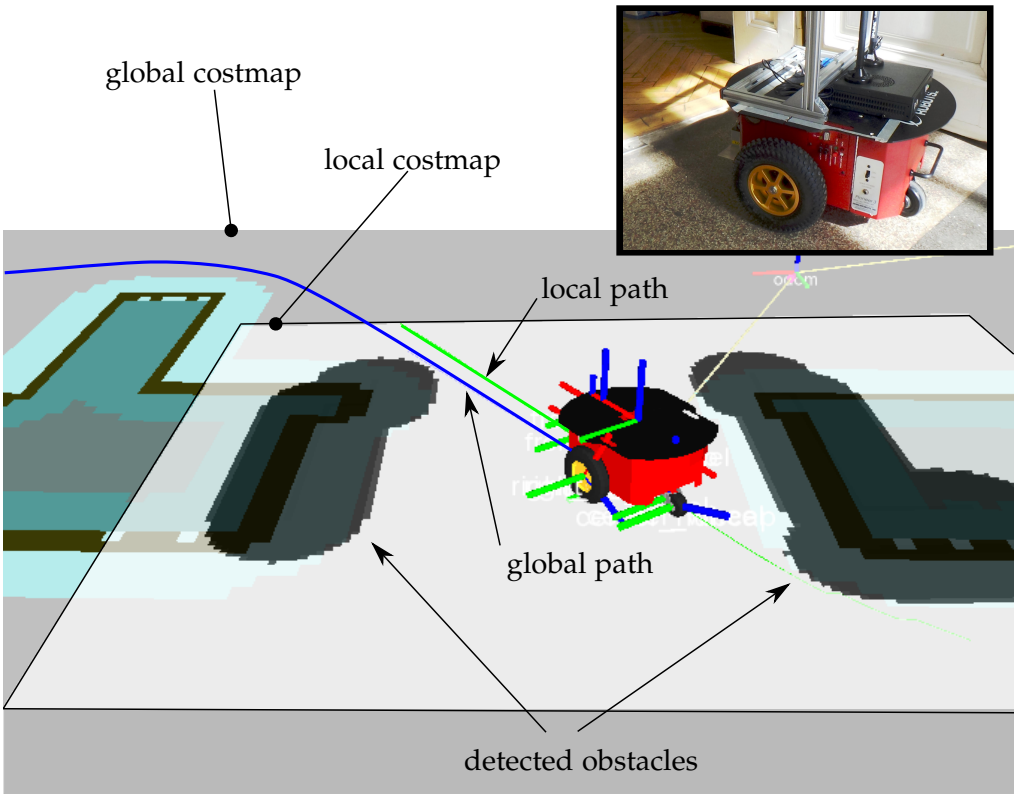


Figure 1: This figure shows a Pionner3DX and its view of the office environment while passing through a door. The blue line shows the global path and the green line the selected trajectory of the local planner.

The major responsibility for local planner is obstacle avoidance. It takes sensor readings of the robot into account and is reactive to changes within the

sensor range. It selects the best values of available motor controls in respect to the kinematic and dynamic constraints of the robot, generating collision free trajectories.

Navigation competence is essential for a broad spectrum of application domains within the field of mobile robotics which are presented below:

SELF DRIVING CARS

Self driving cars (see Figure 2) like Stanley [48] which won the 2005 DARPA (Defense Advanced Research Projects Agency of the United States) Grand Challenge and the Google car [20] are two examples which illustrate the enormous potential of autonomous vehicles. While the former is confronted with rough terrain and maneuvers at high speeds, cars in urban traffic have to be prepared for other vehicles, pedestrians and have to incorporate traffic rules into the navigation task.

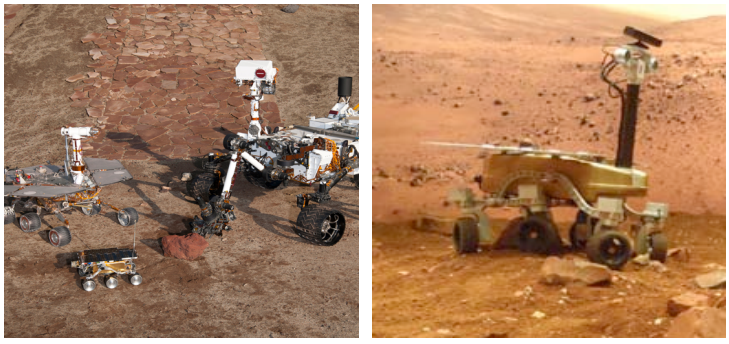
(a) Stanley (taken from [48]). (b) Google car (Credit: Google²).

Figure 2: Self driving cars designed for different environments. On the left the winner of the 2005 DARPA Grand Challenge tackled wreckless driving through the Mojave desert and the right picture shows the new Google car designed for urban traffic.

PLANETARY EXPLORATION

Another example of autonomous vehicles are planetary rovers. Three generations of Mars rovers developed at NASA are shown in Figure 3a. The first Mars rover, Sojourner, which landed on Mars in 1997 as part of the Mars Pathfinder Project was remotely operated from earth. The next generation of rovers, Spirit and Opportunity which landed 2004, did already have an autonomous navigation system. This enabled the robots to avoid hazardous situations without human intervention. The latest rover Curiosity is on its mission on Mars since August 2012 using autonomous navigation to explore mars on safe paths without the need of being remotely controlled by human operators.

An overview of recent developments in the field of planetary exploration including navigation topics can be found in [31].



(a) NASA Mars rover (Credit: NASA JPL-Caltech).

(b) Mawson rover (taken from [35]).

Figure 3: Different Mars rover for planetary exploration. On the left Sojourner, Spirit and Curiosity from NASA which provided rich scientific information exploring Mars and on the right Mawson an academic research project on a training track at the Museum of Sydney.

SEARCH AND RESCUE

Mobile robots provide a useful tool for rescue teams, whenever human intervention is not possible due to imminent risk of health and life, like immediate explosion hazard or threat of nuclear radiation. Figure 4 shows two prominent representatives of search and rescue robots.

(a) Pioneer (Credit: Carnegie Mellon University)

(b) Packbot (Credit: iRobot)

Figure 4: Search and rescue robots are useful tools for human rescue teams. The left figure shows Pioneer at the nuclear disaster site in Chernobyl, the right robot shows a Packbot which operated in the nuclear power plant in Fukushima.

The robot PIONEER sponsored by the US Department of Energy and NASA was the first of its kind to be deployed to the remnants of the nuclear power station in Chernobyl in the Ukraine after the supergau

2 Google self driving car available from http://googleblog.blogspot.co.at/2014/05/just-press-go-designing-self-driving.html

in 1986. The robots mission was to evaluate the sarcophagus which was built to shield off radiation.

In 2011 after an earthquake caused a nuclear disaster at the Fukushima Daiichi power plant in Japan, two PACKBOT robots of the American company iRobot were deployed to perform on site observations.

While the aforementioned robots were remotely operated, autonomous robots are a very active research topic in the academic domain. In the last 5 years more than 90 teams from universities participated in RoboCup competitions and already provide impressive results like Hector [26] the winner of the 2014 RoboCup Rescue League world championship.

ASSISTANCE

Besides robots acting in outdoor environments, assistance robots have to cope with the difficulties involved in operating side by side with humans in indoor environments. Obstacles like desks and chairs located in small corridors and rooms pose a challenging navigation problem. Tour guide robots like Rhino [13] and Robox [42] are especially designed to deal with this kind of setting. Figure 5b shows Robox at the Robotics@Expo.02 event.

Figure 5a shows the robot Hobbit [12][54] which goes one step further in assisting elderly people in their everyday activities. It does not only make an excellent job in dealing with the pitfalls of navigating in indoor environments, in addition it is equipped with the capabilities to identify and to remove obstacles on the floor to provide safe passages for its human users.

(a) Hobbit (taken from [12]) (b) Robox (taken from [42])

Figure 5: Hobbit the *care* robot supporting elderly people and Robox working as a tour guide.

LOGISTICS AND TRANSPORTATION

Automated guided vehicles (AGV) are an essential part in industry to support automated manufacturing. Figure 6a shows a typical industrial AGV from the Austrian company DS-Automotion. The mobile robots follow markers, wires, or magnets in the floor, and make use of lasers and computer vision methods to accomplish navigation tasks. Their main responsibility is to move materials safely and efficient around manufacturing facilities, warehouses (eg. Kiva Systems [19]) or hospitals (eg. HELPMATE [10]).

If transportation by road is not possible a flying drone might do the job. Projects from major companies like Googles *project wing* and Amazons *Prime Air* are working on self flying drones to deliver packages. Figure 6b shows a prototype drone on a test flight.

(a) AGV (Credit: DS-Automotion) (b) Project Wing delivery drone (Credit: KEYSTONE)

Figure 6: Transportation on earth using Automated Guided Vehicles and in the air with the support of self flying drones.

COMMERCIAL

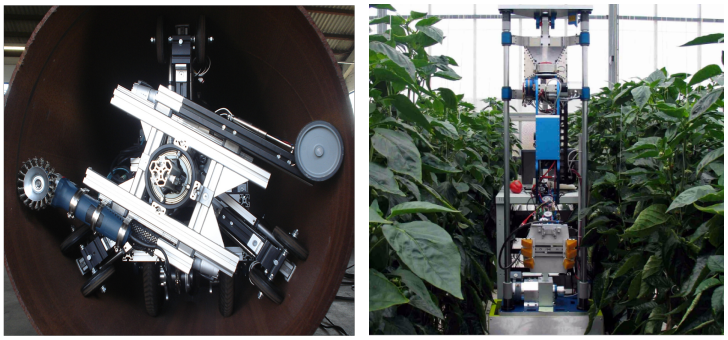
Robots play a vital role in the manufacturing of goods. Besides classical domains such as automotive industry, robots are used for mining, construction and maintenance tasks. For example DeWaLoP, shown in figure 7a, is used for cleaning fresh water pipes, navigating through a 3000 km pipeline network in Vienna [34].

The employment of autonomous robots is also a field of attention in agriculture. Monitoring, harvesting and precision spraying pesticides of crops have to be carefully accomplished without hurting fragile plants. Figure 7b shows a harvesting robot developed at the Technische Universität München moving in a greenhouse [44].

CHALLENGES AND COMPETITIONS

Attractive competitions have the purpose to spur innovation and spon-

2 Project Wing delivery drones https://plus.google.com/+google/posts/TqrsvRyPeNH



(a) DeWaLoP (taken from [34]) (b) Harvesting robot (taken from [44])

Figure 7: The right figure shows an in-pipe maintenance robot, the left image an agricultural robot used for harvesting and spraying of sweet-pepper crops.

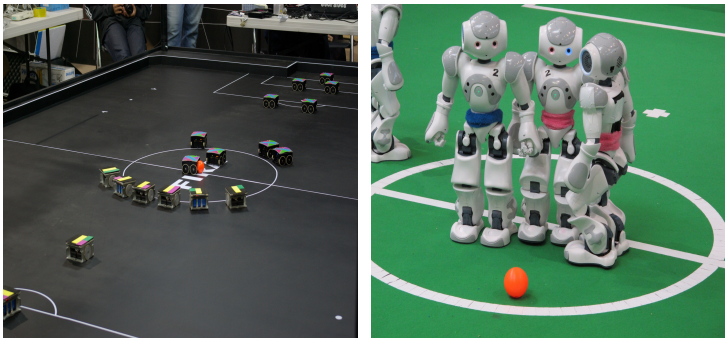
sor research development in the field of robotics . One of the most popular events are organized and sponsored by DARPA. Besides the already mentioned Grand Challenge for wreckless self driving cars, the successor events DARPA Urban Challenge focused on autonomous vehicles moving in urban areas. In 2012 the first DARPA Robotics Challenge took place which provided a platform for wheeled and humanoid search and rescue robots.

World Competitions of soccer playing robots fascinate and motivate thousands of people every year and have a significant impact on the development of innovative methods including navigation and planning for mobile robots. Figure 8 shows wheeled and humanoid robots trying their best at scoring more goals than their opponents. The biggest events are organized by FIRA[3] and RoboCup[4], which provide annual competitions for wheeled and humanoid soccer robots.

All of these examples show that navigation plays an essential role in mobile robotics. Excellent planning algorithms are needed to enable a wide area of applications and allow autonomous and semi-autonomous machines to safely and economically move through in- and outdoor environments. In addition they have to cope with uncertainties induced by sensor noise, the presents of dynamic obstacles and the limits of computational power.

3 Federation of International Robot-soccer Association, founded in 1997. Details can be found at www.fira.net.

4 RoboCup Federation, founded in 1993. Details can be found at www.robocup.org.



(a) MiroSot (taken from www.fira.net)

(b) Standard Platform League (taken from www.robocup2014.org)

Figure 8: The right figure shows competitors of the MiroSot league, the left image humanoid Nao robots playing soccer in the RoboCup Standard Platform League.

1.2 RELATED WORK

Similar approaches using meta-heuristics are mostly present in the field of global planning. Especially population based meta-heuristic algorithms are used to tackle planning problems for single and multiple robots.

In [55] an ant colony optimization (ACO) algorithm is used to find global minimal paths for soccer robots. The algorithm uses a combination of cellular automata and a special designed pheromone model to find an initial global path. In a second step the retrieved path is smoothed and can be used to guide a robot.

In [5] the performance of finding optimal shortest global paths by applying an ACO algorithm is documented. The proposed method uses a special heuristic to set the moving directions of the ants in the system based on shortest distance between nodes in the search graph.

A combination of a local planner based on artificial potential field method and a global planner based on ACO is presented in [36]. Here the pherhormon traces of the ACO is used to prevent local minima in the local planning step.

Several approaches using genetic programming are presented in [30] where solutions to planning problems for autonomous navigation are encoded as chromosomes. According to a fitness function the best chromosomes are chosen to create new solutions via cross over and mutation operations.

An early apporach of using single solution based meta-heuristics in the context of local planning is presented in [41]. Here simulated annealing was applied to avoid the local planner of getting trapped in local minima.

Based on tabu search a sensor based navigation system for mobile robots is presented in [33]. This local planning method utilizes different memory structures to avoid cycling back to already visited regions of the map. To escape

local minima it provides an escaping mechanism by randomly approaching unvisited areas.

The method proposed in this work differs from these works by analyzing and using single solution based meta-heuristics strategies for local path planning only. In addition to the memory structures of tabu search, another contribution is the use of algorithms based on Iterated Local Search (ILS) and Variable Neighborhood Search (VNS). These single solution based meta-heuristics are used to increase the efficiency of a local planner which uses trajectory generation methods like the Dynamic Window Approach (DWA) to find optimal motor controls for the robot.

1.3 GOAL OF THESIS AND SCIENTIFIC CONTRIBUTION

The goal of this thesis is to provide an extension to local planning systems for mobile robots based on trajectory generation and parts of this work were presented at the Austrian Robotics Workshop 2014 [47]. The proposed method applies meta-heuristic search strategies to improve the performance of local planner based on brute force evaluation of a fixed number of trajectories.

This includes the following sub-goals.

COMPILATION OF LOCAL PLANNING APPROACHES
: A detailed overview of local planning and obstacle avoidance methods has to be compiled.

INVESTIGATING POSSIBLE META HEURISTIC ALGORITHMS
: Selection and adapting applicable meta-heuristics to make them usable for trajectory selection.

DESIGN OF BENCHMARK INSTANCES AND SAMPLE PLANNER
: Suitable test instances and a sample planning program which allow to set the focus of investigating the proposed method on the local planning task have to be developed.

IMPLEMENTATION
: The gained data from the benchmark instances and the sample planner are used to implement meta-heuristic local planning in an existing navigation system for mobile robots.

EVALUATION
: Experiments have to be designed and evaluated to document the performance of the meta-heuristic algorithms in comparison to their unaltered counterparts.

1.4 OUTLINE

This work is structured in the following way. Chapter 2 gives an overview of important local planning and obstacle avoidance methods.

Chapter 3 represents the main part of this thesis and introduces meta-heuristic extensions to local planning methods. Starting with an detailed overview of meta-heuristic algorithms, two application based on ILS and VNS are used for trajectory selection of a local planner and described in Section 3.4.

A detailed evaluation of these methods is presented in Chapter 4, using a very simple implementation of a local planner and randomly generated sensor maps. The gathered data is then used to implement a meta-heuristic trajectory selection in a high sophisticated planner and tested using simulation software with a robust physical engine.

Chapter 5 concludes the work of this thesis with a short summary and an outlook on future research directions.

2 LOCAL PLANNING AND OBSTACLE AVOIDANCE

In an environment which is not fully known a priori, the presence of dynamic obstacles, uncertainties concerning the current localization and sensor reading of the robot, local planning methods have to ensure the safety of the robotic vehicle on its way to reach its desired goal configuration.

Taking current sensor readings into account the main task is to safely avoid obstacles while continuously making progress towards the goal. To achieve this task local planner have to make very fast decisions to be able to reactively respond to changes in the environment. These decisions alter the current traveled path by selecting appropriate motion commands for the motor controller. To avoid getting stuck local planning methods include some degree of global knowledge of the environment generated by a global planning component, which is used during the decision process.

The following section describes common approaches of local planning and obstacle avoidance methods.

2.1 BUG ALGORITHMS

A straight forward strategy to generate collision free geometric paths is the family of bug algorithms, inspired by simple path planning strategies from insects.

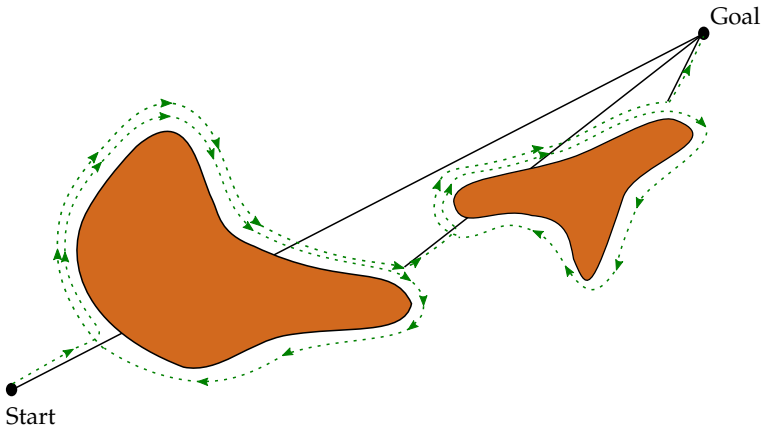
Bug1 and Bug2 are the first mentioned algorithms in this family which are introduced and analyzed in [29]. Both algorithms react to changes in the environment by switching between two behaviors to create a path from the start position to the goal position without hitting an obstacle:

1. **goal-approaching:** Follow the straight line from the current position toward the goal position.

2. **wall-following:** Follow the contour of a detected obstacle.

The differences are in the way the algorithms apply those behaviors. Both algorithms starts approaching the target on a straight line. If Bug1 algorithm approaches an obstacles it follows the whole contour of the obstacles to determine the closest point from the contour to the goal position. From this point the algorithm starts again by applying the goal-approaching behavior. The approach is visualized in Figure 9.

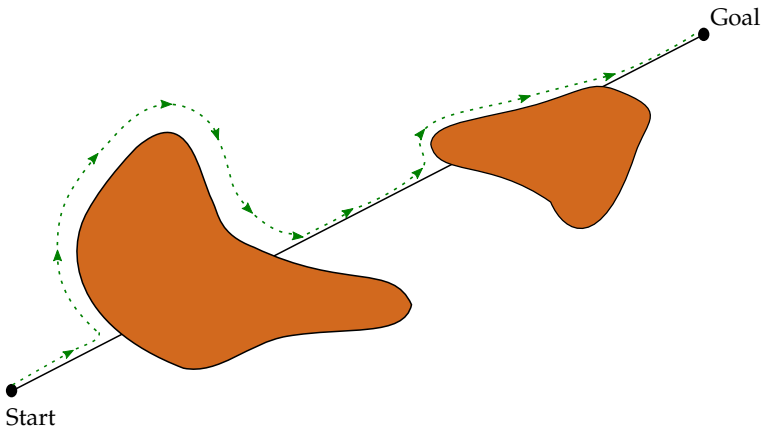
The Bug2 algorithm tries to immediately change from wall-following to goal-approaching behavior as soon as it encounters a point on the contour which



(a)

Figure 9: A path created by Bug1 algorithm.(adapted from [43])

intersects with the straight line connecting starting and goal position, which on average produces shorter paths than Bug1. Figure 10 shows the Bug2 strategy.



(a)

Figure 10: A path created by Bug2 algorithm.(adapted from [43])

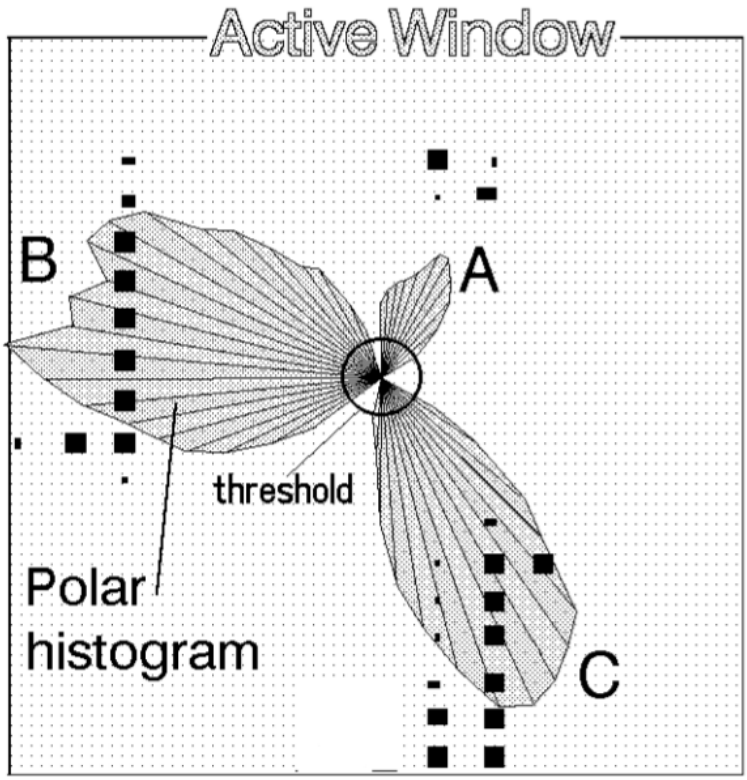
Many improvements on the original approach are presented in the literature (eg.TangentBug [7], PointBug [6]).

2.2 VECTOR FIELD HISTOGRAM (VFH)

Vector Field Histograms [3] simplify the representation of the environment in a 2 step approach to allow for fast calculation of good directions for holonomic robots. In a first step a small local two-dimensional Cartesian *histogram grid* is created. Each cell of the grid is incremented if an obstacle is reported from the sensors of the robot. Cells with higher values indicate a higher probability of

an obstacle at this location, which account for the uncertainty of the sensor readings.

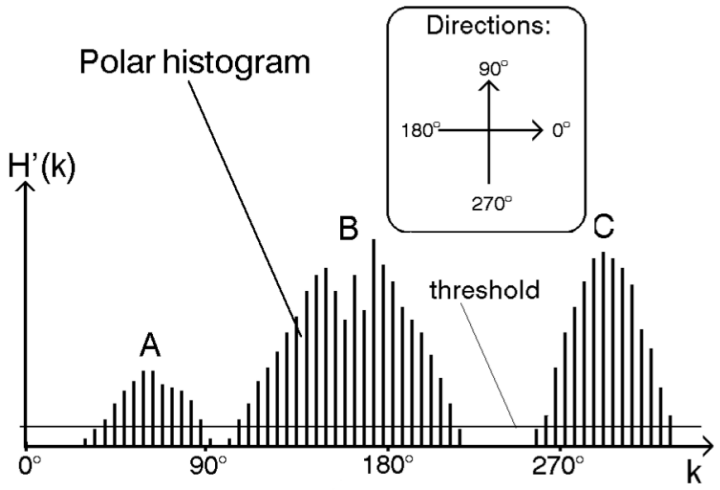
In a second step the grid is further reduced into a *polar histogram* with angles at the x-axis which indicate an obstacle. The angles are divided into a discrete set of sectors and each sector obtains an obstacle density value based on the values from the histogram grid. A threshold value is used to identify valleys in the polar histogram which are used to find gaps large enough for the robot. The identified valleys are then evaluated using a cost function. Figure 11 shows an example of the histogram grid, and figure 12 the corresponding polar histogram.



(a) Histogram grid

Figure 11: This figure shows a histogram grid with three obstacles A,B and C. Darker cells indicate a higher probability of obstacles. From this information a one dimensional polar histogram is derived for further processing. (taken from [3])

Improvements of this method can be found in the VFH+ [50] and VFH* [51] method. Here the second step yields an even simpler *binary* polar histogram, where 1 indicate a blocked and 0 a free sector. In addition some of the kinematics and dynamics are taken into account and the purely local approach is enhanced by adding information of a global path planning component using A* algorithm.



(a) Polar histogram

Figure 12: This figure shows the corresponding polar grid from the histogram grid in Figure 11. Large enough gaps in the polar grid are used to find the optimal steering directions for the robot.(taken from [3]). The horizontal axis shows the direction angles and the values of the vertical axis indicate the probability of an obstacle in the corresponding direction.

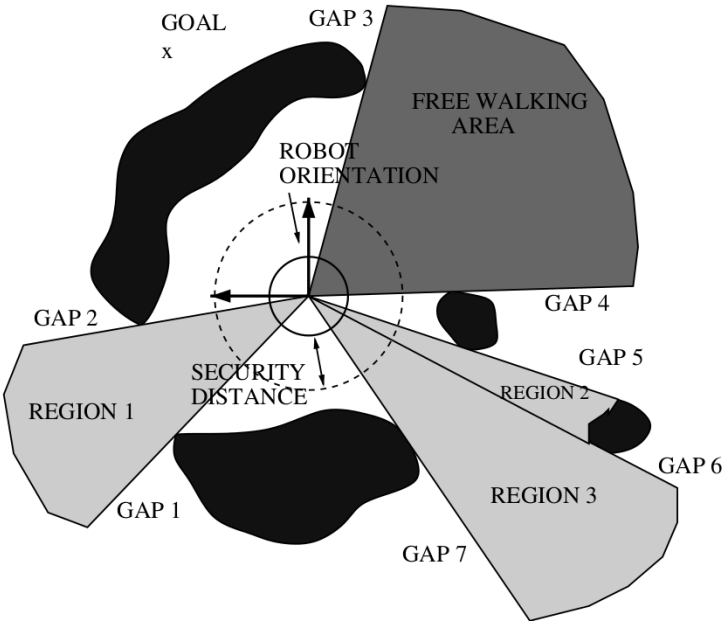
2.3 NEARNESS DIAGRAMS (ND)

Another approach that makes use of graphic representation as a diagram is the Nearness Diagram method presented in [37] and its refinement in ND+ [39]. The surrounding of the holonomic robot, approximated as a circle, is divided into a set of different sectors similar to the VFH method.

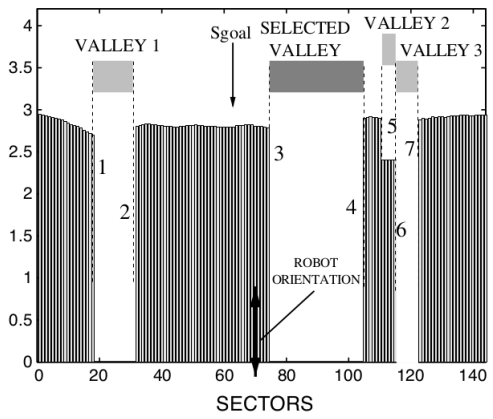
Sensor information about obstacles is used online to generate two different kind of diagrams. The PND represents the nearness to obstacles to the center point of the robot, and the RND the nearness to obstacles to the robots boundary. These diagrams are used to categorize different regions in the surrounding of the robot. RND is used to guarantee safety constraints for the robot, while PND is used to identify valleys which are used for further evaluation. The region which is closest to the goal and reachable by the robot is selected as the *Free walking area*. Figure 13 shows identified regions and the corresponding diagrams.

The best actual steering direction is evaluated using a binary decision tree representing a set of predefined *situation* and *action* pairs. The values of RND and PND are applied as input and are processed by the branching rules of the tree.

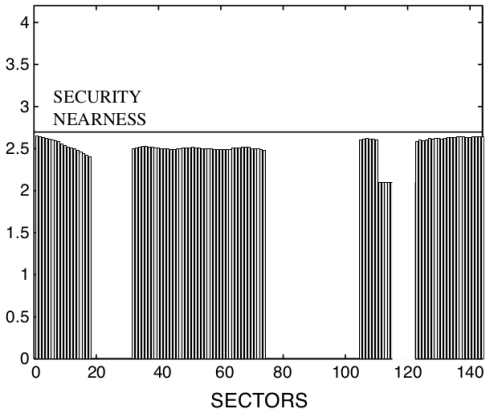
Improvements to this mehthod are the Generald Nearness Diagrams (GND) method presented in [38] which adds global information to the evaluation process. The Smooth Nearness Diagrams (SND) introduced in [9] improves



(a) Regions in the surrounding of the robot.



(b) PND diagram



(c) RND diagram

Figure 13: Figure (a) shows identified regions using the PND diagram in (b) and RND diagram in (c).(taken from [37])

on the ND+ method by providing a unique motion law for all situations and incorporating all visible obstacles surrounding the robot, creating smoother paths for the robot.

2.4 CURVATURE VELOCITY METHOD (CVM)

The Curvature Velocity Method in [46] describes an obstacle avoidance method which considers kinematic and dynamic constraints of the robot and the environment. These constraints are added to a *velocity space* which consists of *translational velocity* v and *rotational velocity* w. One basic assumption of this approach is that the robot can only travel along circles with curvature $c = w/v$. This approximation neglects acceleration issues and restricts the robot motions to constant velocities over a given time horizon.

Another simplification is the representation of obstacles as circles, which is used for fast transformation of the obstacles into the velocity space. All curvatures which do not hit obstacles and adhere to the kinematic constraints of the robot are then evaluated using a cost function. Since approximation techniques like simulated annealing to maximize the cost function using the whole velocity space did not succeed, the velocity space is divided into finite sets of curvature intervals. The objective function is then evaluated over all curvature intervals.

A problem of this method arises when confronted with narrow passages which are perpendicular to the robots heading, which might lead to missing shorter paths. This problem is solved by the Lane Curvature Method [25], by dividing the workspace into finite number of lanes. The best angular velocity for changing lanes is evaluated using the CVM method.

2.5 DYNAMIC WINDOW APPROACH (DWA)

A well known method for local planning is the Dynamic Window Approach proposed in [13]. The method samples the *velocity space* (v, w) of the robot, where again v is the *linear velocity* and w the *angular velocity* of the robot, to create a set of feasible trajectories. The space is reduced to a *dynamic window* which contains the reachable minimal and maximal velocity in one control cycle, taken the acceleration limits of the robot into account.

Obstacles are transformed into the velocity space using a distance function. Figure 14 shows the obstacles in velocity space and the corresponding dynamic window.

For a fixed amount of velocity samples the corresponding trajectories are created using a predefined granularity by performing forward simulation for a short period of time, starting at the current position and speed of the robot. The trajectories which stop safely before an obstacles are called *feasible*. Evaluating all *feasible* trajectories with respect to a weighted cost function (cf. Equa-

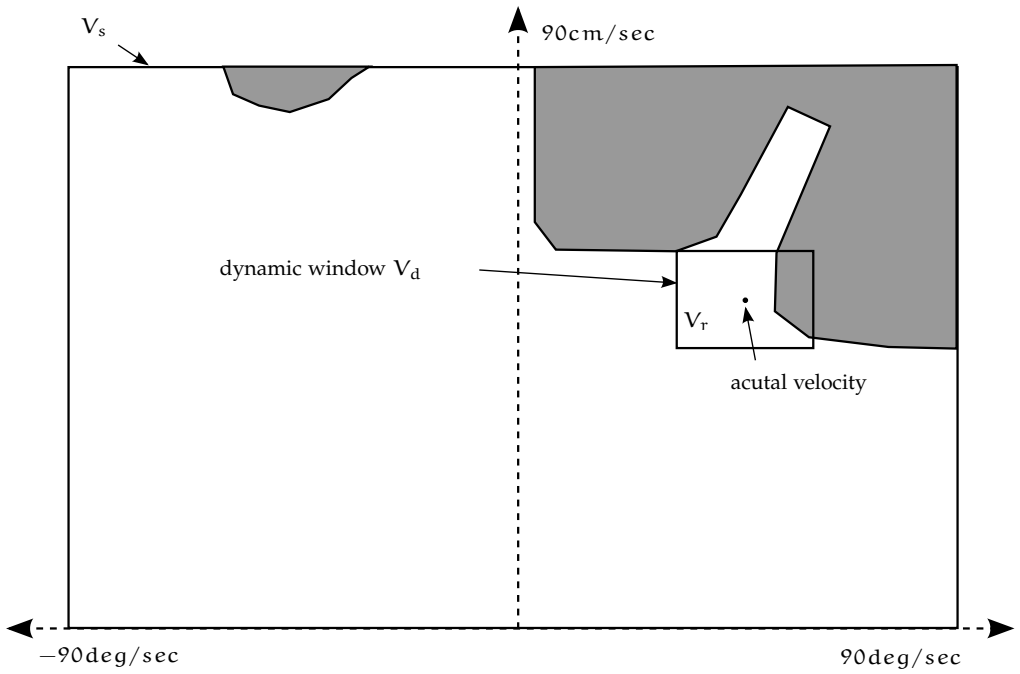


Figure 14: This Figure shows a dynamic window V_d of a robot together with obstacles transformed in a velocity space V_s. Only the velocities within the small area around the current robots speed V_r are used for evaluating trajectories by a cost function. The best velocities are forwarded to the robots motor controller. (reproduced from [13])

tion 1) identifies the best velocity tuple, which is then forwarded to the motor controller.

$$f_c(v, w) = \alpha f_a(v, w) + \beta f_d(v, w) + \gamma f_v(v, w) \tag{1}$$

The function $f_a(v, w)$ judges the angle between the robots heading and a given goal position. It is maximal if the heading is a straight line to the goal. The distance to the closest obstacle is calculated in the function $f_d(v, w)$. The function $f_v(v, w)$ takes the forward velocity into account and rewards faster movements of the robot.

In this method the obstacles are not approximated by circles, instead small lines are used to account for a more accurate representation of obstacles. The robot is still approximated as a circle. The original method does not use a global plan to guide the robot, so without further changes it is subject to get captured in local minima.

The close relationship of the DWA to Model Predictive Control (MPC) has been shown in [40]. In this paper the DWA is reformulated as a problem of MPC which together with a navigation function has the proven property to converge to a global optimal solution.

Other applications of this approach in recent planning systems adapt the corresponding cost function. The excellent `move_base`[1] motion planning framework introduced in [32] implements within the navigation stack of Robot Operating System (ROS) a local planner which uses a global plan as a guide, and uses a polygon to model the robot outline.

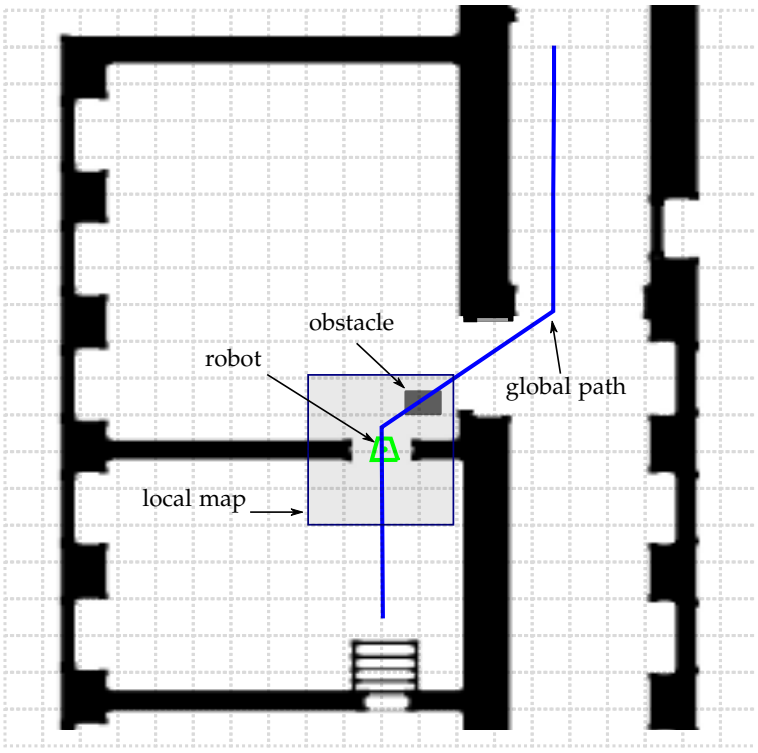
One implementation is based on DWA. There is also the option to use Trajectory rollout [16] as a local planner which is very related to the DWA, but in contrast improves in simulating the robots trajectory by accurately applying acceleration limits over the whole simulation time. The cost function maximizes characteristics like proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Furthermore a number of escaping strategies try to avoid the vulnerability to local minima.

Collision detection and cost calculation is performed by using the footprint of the robot following the calculated trajectory. Hence the discretized footprint, which is usually given as a simple polygon, is projected on the costmap. Bresenham's Line algorithm [4] is used for ray-tracing the contour of a robot in the discrete workspace. Figure 15a shows the global view of the planning task. In Figure 15b the corresponding local view is depicted, including all sampled trajectories which are evaluated using a local costmap.

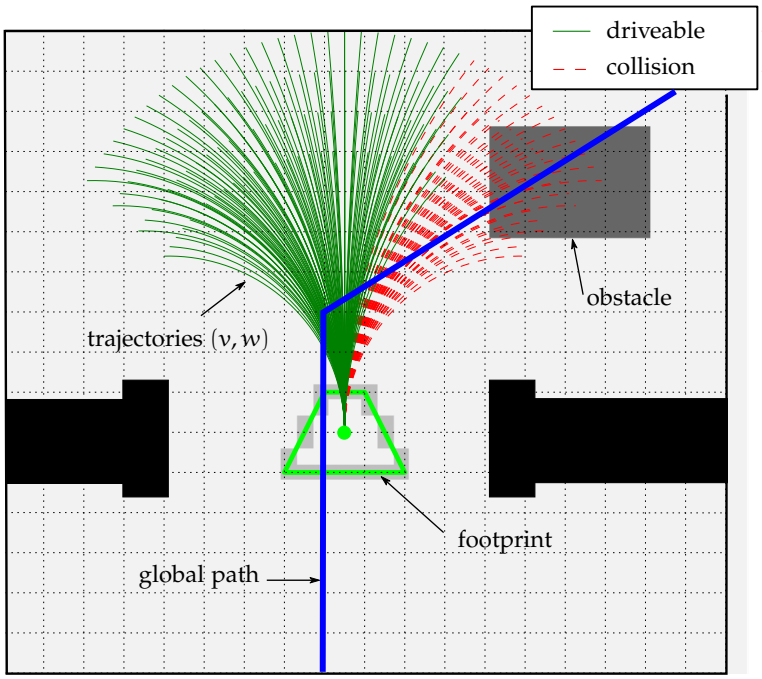
Concerning the optimization of the used cost functions the most common approach for DWA and related local planner is to evaluate all possible trajectories in a reduced discrete velocity space. Examples of this approach can be found in [24][32][45].

The proposed method extends the DWA approach, by using approximation algorithm to maximize the cost function in a discrete representation of the velocity space.

1 move_base planning framework: http://wiki.ros.org/move_base



(a) Path planning in global costmap.(taken from [47])



(b) Trajectory generation for linear, and angular velocities (v, w) in local costmap guided by global path.(taken from [47])

Figure 15

Part II

APPROACH

3 META HEURISTIC LOCAL PLANNING

This thesis proposes an approach to enhance certain types of local planning systems. Examples of such local planners are the aforementioned Dynamic Window Approach and Trajectory Roll-out planner, which were introduced in Section 2.5.

The idea is to apply well established meta-heuristic search algorithms to find good trajectories given a sample of different applicable velocities, a robotic motion model and a given time period for simulation. This work introduces trajectory selection extensions using a combination of meta-heuristic algorithms.

- **ILS-Tabu:** Based on Iterated Local Search and the use of a simple memory
- **VNSB-Tabu:** Based on Variable Neighborhood Search and the use of a simple memory and Best Improvement heuristic.
- **VNSF-Tabu:** Based on Variable Neighborhood Search and the use of a simple memory and First Improvement heuristic.

The proposed methods are tested and refined and implemented into a robotic navigation system which already supports implementations of a local planner based on Dynamic Window Approach and Trajectory Roll-out.

- **VNS-ROL:** Refined VNSB-Tabu with short term memory for Rollout planner
- **VNS-DWA:** Refined VNSB-Tabu with short term memory for DWA planner

The next section provides background information of meta-heuristic search. Following well known classifications of meta-heuristics (see [2], [1]) an introduction of *single solution based* meta heuristics is given, which are the main source for the proposed approach.

This chapter is then concluded by a detailed description about how meta-heuristics are used to extend local planners in the trajectory selection process. Furthermore all relevant information concerning the local planners is provided.

3.1 META-HEURISTIC SEARCH

The family of meta-heuristic algorithms is known to be successful in solving optimization problems, and are heavily used to solve hard problems whether

they are *discrete* (Combinatorial Problems: e.g. Traveling Salesman Problem (TSP), MAX-Sat problem, Nurse Rostering problems (NRP), Classical Vehicle Routing Problem (CVRP),...), or *real-valued* (Continuous Optimization Problems: e.g. Pooling problem, Continues Min-Max problem,...).

In general an optimization problem and its solutions can be defined as follows [2]:

Definition 1. An optimization problem $P = (S, f_c)$ can be defined by:
- a set of *variables* $X = \{x_1, \dots, x_n\}$;
- a set of *domains*, defining the values the variables can take $D_1, \dots, D_n$;
- possible *constraints* on the variables
- a *cost function* (or more general an *objective function*) f_c which has to be minimized, where $f_c : D_1 \times \cdots \times D_n \to \mathbb{R}^+$; (which is the same as maximizing $-f_c$)

The set of all possible assignments $S = \{s = \{(x_1, v_1), \dots, (x_n, v_n)\} \mid v_i \in D_i\}$ to the variables in X is called the *search* or *solution space*. A continues optimization problem is given if $S = \mathbb{R}^n$, otherwise if S is finite but large set a combinatorial optimization problem is given [21].

Solutions which fulfill all constraints are called *feasible* solutions, solutions violating constraints are called *infeasible*. It is not necessary to allow only feasible solutions in the search space, as accepting infeasible solutions during the search process can improve the overall solution quality and performance of the algorithm.

The solution $s^* \in S$ of an optimization problem with minimum cost is called a *global* optimal solution.

Definition 2. A *global* minimal solution is a solution $s^* \in S$ such that $f_c(s^*) \leqslant f_c(s) \forall s \in S$ [2].

Since there might be more than one optimal solution $S^* \subseteq S$ is the set of *globally* optimal solutions.

Due to large solution space, or hard run time constraints finding an exact global optimal solution is not always feasible. Instead a *good* approximate solution, with the help of an heuristic is used and optimality is traded off with good performance.

One basic heuristic approximation method is local search. Given an initial solution local search tries to iteratively replace this solution with a better one in a defined neighborhood of the current solution.

Definition 3. A neighborhood structure is a function $N : S \to 2^S$ that assigns to every $s \in S$ a set of neighbors $N(s) \subseteq S$. $N(s)$ is called the neighborhood of s [2].

Defining the neighborhood function has an important influence on the good performance of many meta-heuristic algorithms. A neighborhood may be induced from metric functions introduced into S given some notion of nearness within a given starting point.

The steps from one local solution to another local solution within a neighborhood is called a *move*. Neighborhoods can also be induced by the set of applicable moves given a current solution [14].

Minimal solutions within a given neighborhood structure are defined as *local* minimal solutions.

Definition 4. A *local* minimal solution w.r.t. a neighborhood structure N is a solution $\hat{s}$ such that $\forall s_n \in N(\hat{s}) : f_c(\hat{s}) \leqslant f(s_n)$ [2].

A visualization of the concepts of local vs. global minimal solutions is visualized in Figure 16.

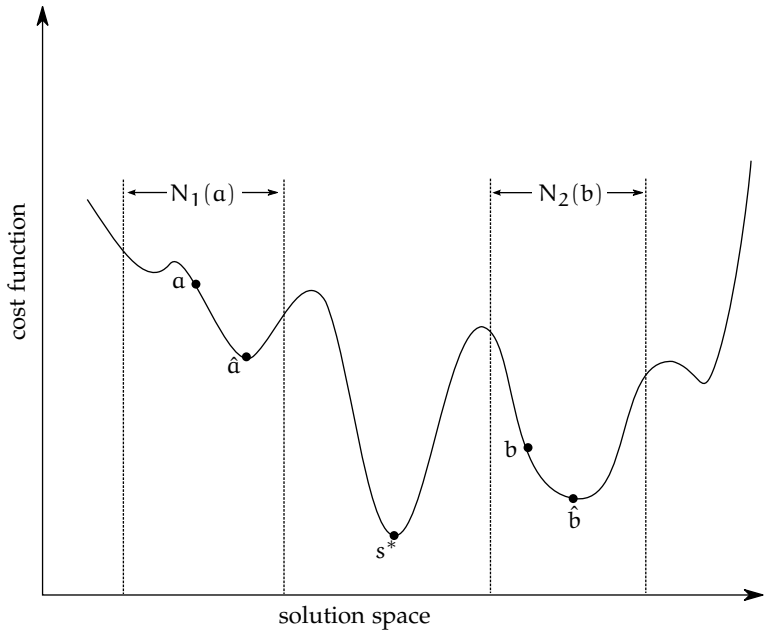


Figure 16: This figures shows different kind of solutions to an optimization problem. The solutions $\hat{a}$ and $\hat{b}$ are local minimal solutions in the neighborhood $N_1(a)$ and $N_2(b)$ respectively, whereas the global minimal solution is s^*.

The main idea of meta-heuristic algorithms is to combine heuristic methods with higher level strategic components to guide the search in an effective and economic manner. The name was introduced by Fred Glover who described his tabu search approach "as a *meta-heuristic* superimposed on another heuristic "[17].

How successful a meta-heuristic performs depends largely at how good it is to balance two important objectives during a search:

INTENSIFICATION
: The search focuses on regions with high quality solutions, which are identified using knowledge gained during the search process.

DIVERSIFICATION
: The search escapes a local minimal solution and visits unexplored regions of the search space.

3.2 SINGLE SOLUTION BASED META-HEURISTICS

As the name suggests these meta-heuristic approaches operate on improving a single solution during the search. Starting from an initial solution the search describes a single trajectory in the search space, when moving from one solution to another. Usually these methods incorporate local search as a component, and additional strategies to escape from local minimal solution.

3.2.1 *Local Search*

The simple local search (LS) (cf. Algorithm 1) finds a local optimum according to a cost function $f_c(x)$ in a region around an initial solution in the solution space, defined by a neighborhood structure $N(x)$.

Algorithm 1 Local Search (LS)

```
1: x ← initial solution
2: repeat
3:     x ← SELECT(x)
4: until no improvement
```

Starting from an initial solution it iteratively selects elements from the neighborhood which improve the current solution. The search stops at a local minimal solution if there are no improving solutions which can be selected.

A frequently used selection functions is *Best Improvement*(or steepest descent) (cf. Algorithm 2), which exhaustively looks at each neighbor to select the best solution.

Algorithm 2 Best Improvement

```
1: function BESTIMPROVEMENT(x)
2:     i ← 0
3:     repeat
4:         i ← i + 1
5:         x_i ∈ N(x), retrieve the i-th neighbor
6:         if cost(x_i) < cost(x) then
7:             x ← x_i
8:         end if
9:     until i = |N(x)|
10:    return x
11: end function
```

Another popular selection function is *First Improvement* (or first descent) (cf. Algorithm 3), which looks at the neighbors in a specific order and returns as soon as a better neighbor is found.

Algorithm 3 First Improvement

```
1: function FirstImprovement(x)
2:     i ← 0
3:     repeat
4:         i ← i + 1
5:         x_i ∈ N(x), retrieve the i-th neighbor
6:     until i = |N(x)| or cost(x_i) < cost(x)
7:     return argmin(cost(x_i) , cost(x))
8: end function
```

For selecting the next suitable neighbor it is also possible to choose a neighbor at random, or choose the best neighbor among a random sample taken from the current neighborhood. These strategies could be used if the number of neighbors is very large, or their evaluation is computational expensive.

3.2.2 *Tabu Search*

An often successful meta-heuristic strategy is Tabu Search [17], which uses memory structures called *tabu lists* to guide the search process.

The memory keeps track of complete solutions (*explicit* memory), moves, partial solutions, or other solution attributes (*attributive* memory), which are used to prohibit moves during the exploration of the search space for a given amount of time - they are *tabu*[1]. In Algorithm 4 the basic form of this approach is presented.

Algorithm 4 Tabu Search

```
1: Tabulist ← 0
2: x ← initial solution
3: repeat
4:     X' ← Neighborhood(x) ∉ Tabulist
5:     x' ← best solution in X'
6:     Tabulist = Tabulist ∪ {x'}
7:     x ← x'
8:     if x is overall best solution then
9:         store x as best solution
10:    end if
11: until stopping criteria satisfied
```

1 The word tabu (or taboo) stems from an Polynesian language and indicates things which may not be touched [18]

The main usage of tabu lists is to prevent cycling back to recent solutions, and prevents from searching the same region of the search space over and over again. This *short term memory* approach contributes to the *diversification* of the search, since the neighborhood of solutions is changed dynamically according to their tabu status.

Usable other types of memory structures are the following (presented in [18]):

RECENCY-BASED MEMORY
: Recording when an attribute or solution was used in the search process.

FREQUENCY-BASED MEMORY
: Recording how often an attribute or solution was used during the search process.

QUALITY-BASED MEMORY
: Recording attributes or solutions which lead to, or are common to good solutions.

INFLUENCE-BASED MEMORY
: Recording information of attributes or solution considering the impact of solution quality and structure.

According to the *short term memory* analogy of the human memory, tabu lists can be furthermore classified in *intermediate term memory*, and *long term memory* strategies [14].

SHORT TERM MEMORY
: Used for *diversification* of the search. Keep track of a small amount of recent moves, thus avoid cycling back to recent solutions.

INTERMEDIATE TERM MEMORY
: Used for *intensification* of the search. For example use a recency-based memory, which records good solution and restart the search from these elite solutions.

LONG TERM MEMORY
: Used for *diversification* of the search. For example implementing a frequency based memory recording the total number of iterations in which solutions or their components have been used in the current solution. Restart the search using solutions which have a lower number of iterations.

An important role in controlling the *diversification* of the search is the length of tabu lists (*tabu tenure*). As a general rule the longer the tabu list the higher the diversification effect, since preventing a larger number of candidate solutions in a neighborhood drastically changes the solution landscape [2].

Restricting the tabu lists length is usually achieved by implementing it as cyclic lists, and deleting e.g. the oldest item as soon as a new item becomes tabu. The tabu tenure can both be fixed or may be changed during the search.

To avoid the tabu lists being too restrictive, and possibly loose good solutions, the tabu status of solutions may also be overridden using an *aspiration criteria*. One example of such an criteria is to always allow solutions which are overall best, even if they are tagged as tabu.

A big advantage of this method is, that it can be easily combined with other meta-heuristic algorithms.

3.2.3 *Iterated Local Search*

Iterated Local Search provides a highly general scheme for search space exploration. It emphasizes the idea of treating the local search procedure as a black box, which in the ideal case incorporates all problem specific knowledge. A detailed description of Iterative Local Search including the basic algorithm (cf. Algorithm 5) can be found in [28].

Starting from an initial solution, the algorithm iteratively calls the embedded *local search* procedure. In each iteration the current solution might be *perturbed* by changing parts of the solution. The following *local search* takes this altered solution as a new starting point and if it satisfies an *acceptance criterion* the search restarts with *perturbating* this new solution.

The basic steps of ILS are visualized in Figure 17

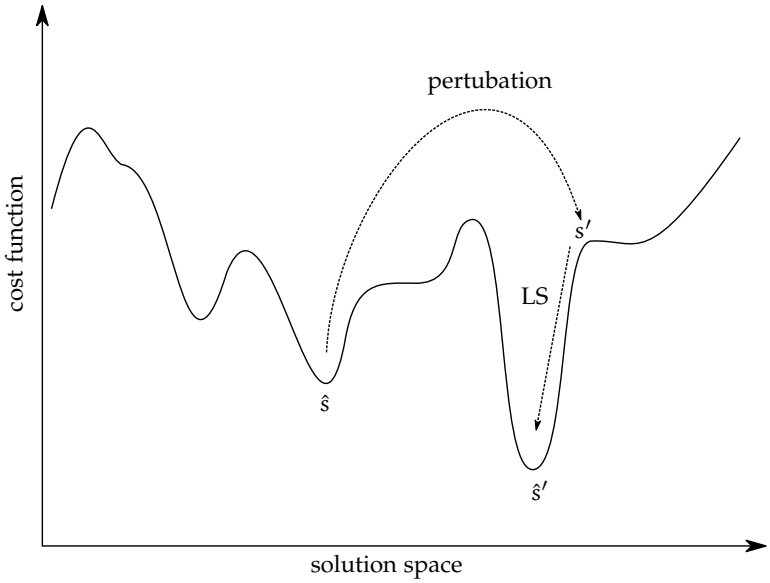


Figure 17: Perturbation of a local minimal solution $\hat{s}$, leads to an intermediate solution s'. A local search is applied and a new local minimal solution $\hat{s}'$ is found. (reproduced from [2])

The *strength* of the perturbation is defined as the number of changed solution components. If the perturbation strength is large, the *diversification* effect is large and the procedure is similar to a random restart method. On the other hand if the strength is small, perturbation favors the *intensification* of the search. The strength may be fixed but may also be adapted during the search [28].

The *acceptance criterion* is the other mechanism in ILS which controls the interplay between *diversification* and *intensification* of the search. On one extreme accepting only better solution favors intensification, on the other extreme accepting all solutions without regards to its costs has a high diversification effect. Every intermediate mechanism and incorporating adaptive strategies may be used in designing acceptance tests.

Another important component is the use of short or long term memory concepts, which may be included in form of a *history* of the search. The *history* is used to steer the perturbation step and the acceptance test (compare to Tabu Search memory structures in Section 3.2.2). As a simple example the history counts the iterations and restarts the algorithm after reaching a specific amount of iteration.

Algorithm 5 Iterative Local Search (ILS)

1: $x_0 \leftarrow$ initial solution
2: $x^* \leftarrow$ LocalSearch(x_0)
3: **repeat**
4: $\quad x' \leftarrow$ Perturbation(x^*,history)
5: $\quad x^{*\prime} \leftarrow$ LocalSearch(x')
6: $\quad x^* \leftarrow$ AcceptanceCriterion(x^*,$x^{*\prime}$,history)
7: **until** stopping criteria satisfied

3.2.4 *Variable Neighborhood Search*

Instead of using a fixed neighborhood, the Basic Variable Neighborhood Search (cf. Algorithm 6) as presented in [21] uses a neighborhood structure of possibly nested neighborhoods (cf. Equation 2), which together are guaranteed to explore the whole solution space.

$$N_k(x) = N_0(x), N_1(x), \ldots, N_{k_{max}}(x) \tag{2}$$

A specific neighborhood structure determines the topology of the search space within this neighborhood, or in other words a neighborhood specific landscape [2].

Different neighborhood structures yield different landscapes. Furthermore a local minimal solution within one landscape, does not have to be a local minimal solution in another, and a search procedure will find a better local minimal solution (see Figure 18).

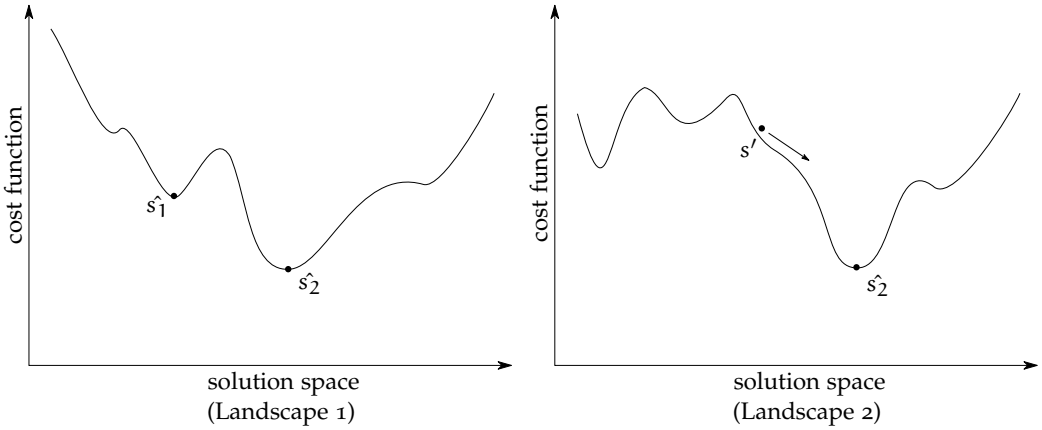


Figure 18: This figures shows two different landscape from different neighborhood structures. On the left image the local search stops at the local minimal solution $\hat{s}_1$. On the right the local search can proceed to a better local minimum $\hat{s}_2$. (reproduced from [2])

In the *shaking* phase the algorithm chooses a random solution of the current neighborhood to avoid getting captured in a local minima.

If the solution found by local search does not improve the next neighborhood will be considered (cf. Algorithm 7). The selection of this neighborhood might be deterministic, but also other schemes are possible. For example one could randomly choose the next neighborhood, or use additional parameter to adjust the step sizes for the *neighborhood change* dynamically.

Variants of this basic scheme are obtained by removing the random shaking phase, which results in deterministic descent method called *Variable neighborhood descent* (VND). Omitting the local search step results in the *Reduced VNS* (RVNS) method, which is a purely stochastic method.

Algorithm 6 Variable Neighborhood Search (VNS)

```
1:  function VNS(x, k_max)
2:      repeat
3:          k ← 1
4:          repeat
5:              x' ← Shake(x, k)
6:              x'' ← LocalSearch(x')
7:              NeighborhoodChange(x, x'', k)
8:          until k = k_max
9:      until stopping criteria satisfied
10: end function
```

Algorithm 7 Neighborhood Change

```
1: function NeighborhoodChange(x, x′, k)
2:     if cost(x′) < cost(x) then
3:         x ← x′
4:         k ← 1
5:     else
6:         k ← k + 1
7:     end if
8: end function
```

3.2.5 *Guided Local Search*

Changing the neighborhood of solutions, as in the aforementioned algorithms like ILS and VNS, is not the only mechanism to escape local minimal solutions. A very different approach in guiding the search for optimal solutions is Guided Local Search (GLS), and was introduced by Tsang and Voudouris [49][52].

In this approach diversification is achieved by changing the cost function itself. The neighborhood structure stays fixed. For intensification a local search method is performed in each iteration. In Figure 19 the gradually change of the cost function which may lead to new local minimal solutions is visualized.

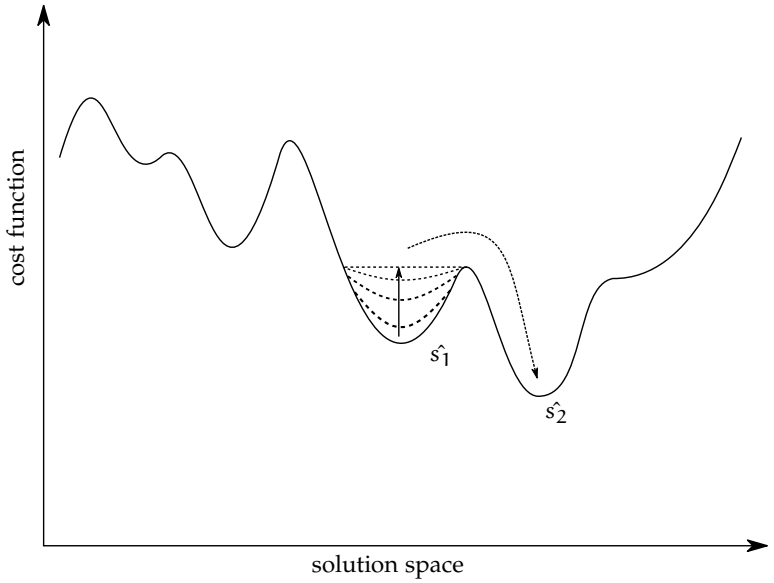


Figure 19: Escaping a local minimal solution by gradually increasing the cost function. (reproduced from [2])

Altering the cost function is based on penalizing the presence of *solution features* which are distinguishing properties of different solutions. The effect

of the *penalty parameters* on the cost function is in addition balanced via a *regularization* parameter.

3.2.6 *Simulated Annealing*

Simulated Annealing (SA), which is one of the first algorithms considered a meta-heuristic, concludes this presentation of single solution based meta-heuristics.

It was introduced in [23], and provides an explicit *diversification* mechanism to avoid getting trapped in local minimal solutions. This is achieved by allowing moves to worse solutions during a local search with a certain probability (metropolis criterion, see 3).

$$P(\mathrm{accept}(x')) = \exp\left(\frac{-|f_c(x') - f_c(x)|}{T}\right) \tag{3}$$

The interplay between *intensification* and *diversification* is controlled by the temperature parameter T. A high temperature yields a high probability for moving to the new solution. On the other hand a low temperature makes it less probable to leave the current neighborhood.

Decreasing the temperature during the search is called *annealing*, referring to the intuition of physical annealing processes, which sets materials in a stable (optimal) state by reducing their temperature.

Beside this basic form of a *cooling schedule* oscillating schemes can be implemented (*reheating*) which cycle between *intensifiction* and *diversification* phases.

3.3 SUMMARY

The presented meta-heuristics provide a powerful tool for general purpose optimization. This work restricts the collection of investigated algorithms which operate on a single solution. In contrast to the aforementioned methods, population based meta-heuristics operate on a set of solutions. Instead of a single trajectory, the search process describes "the evolution of a set of points in the search space "[2]. Examples of such algorithms include methods based on Evolutionary Computation, Ant Colony Optimization, and Artificial Immune Systems. For an in depth presentation of current meta-heuristic methods see [15].

The next section covers the approach to apply meta-heuristic search in the context of local planning.

3.4 TRAJECTORY SELECTION USING META-HEURISTICS

One basic step in creating a good local planner for mobile robots is the aforementioned trajectory selection. Especially forward movements are of particular interest, since their evaluation and collision tests are costly operations. Only if forward movement does not yield valid movements, other strategies have to be applied, which provide escaping mechanisms, like turning around for relocation, or slow random backward movements to help the robot evade situations where it gets stuck. The use of escaping mechanisms should be limited, since they are only relevant if the usual planning step fails.

In this thesis the focus lies on improving the trajectory selection of local planners using meta-heuristic algorithm.

Instead of following an exhaustive generate and test procedure on all velocity samples, meta-heuristic algorithms are used to increase the search performance.

Increasing the performance of the local planner has an important impact on the resulting path and behavior of the robot. It allows for increasing the following properties of local planner:

REACTIVITY
: Clearly one benefit of decreasing the time the local planning steps take is its increased reactivity. The faster and hence more often the local planner can be invoked to adapt to recent sensory information.

RESOLUTION
: The performance gain can be used to increase the resolution of cost-maps and collision test of the robot.

SIMULATION-TIME
: Performing longer simulation periods results often in smoother paths since the robot can react early to possible obstructions. In addition it might also find shortcuts at an earlier stage, which results in a shorter path.

TRAJECTORY-NUMBER
: Another very important quality benefit is the number of trajectory samples the local planner can use for making a decision. A higher number of trajectories increases the granularity of the cost function it optimizes. In the end this yields a smoother and more effective path (see Figure 20).

The choice of using single solution based meta-heuristics (see Section 3.2) was driven by the properties these algorithms provide. They are *general applicable* to optimization problems, since no domain specific knowledge is necessary, or can easily be captured using only the local search procedure. Furthermore they are easy to implement and incorporated into existing software solutions.

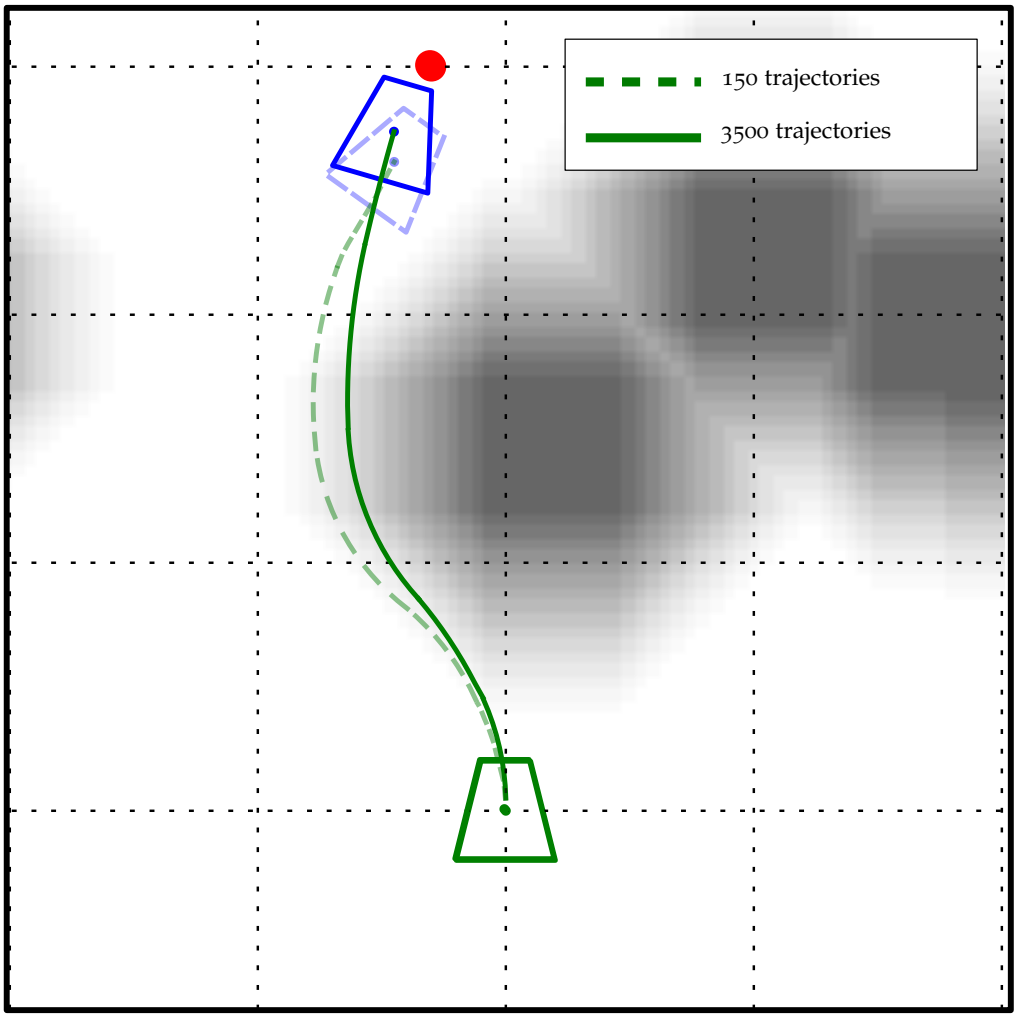


Figure 20: Comparison of two paths generated using different amounts of trajectories. The dotted path was generated with 150, and the other one with 3500 trajectories. The higher amount of sampled trajectories results in a smoother and more effective path.

Looking at the high number of parameters used for configuring current navigation systems the *configuration process* is very tedious. Hence introducing lots of new parameters does not make it easier to find suitable configurations. Meta-heuristics are well suited for this task because they use only few parameters to achieve good performance.

In order to use meta-heuristics the local planning problem has to be formulated according to the definitions given in Section 3.1.

Since this work deals with the navigation of non-holonomic robots moving on a 2D-plane workspace, the *input variables* are all tuples of forward and angular velocities (v, w) the robot is capable. The *domain* of the variables is a discrete set of the real valued search space induced by using a fixed stepping size and selecting a specific amount of trajectories.

Forward and angular velocities (v, w) are furthermore *constrained* within given limits $v_{min} \leqslant v \leqslant v_{max}$ and $w_{min} \leqslant w \leqslant w_{max}$.

The optimization problem is given as a function over the control inputs (velocity tuples) for the motor controller. The costs are calculated using a simulation of the robot with the given input velocity tuple and projecting the outline of the robot onto a cost-map.

A generated cost-map together with a visualization of consecutive local path planning steps in a simulation is shown in Figure 21.

Solutions which do not exceed a given cost threshold are *feasible* and free of collision. On the other hand solutions which reach a given maximum cost value indicating a collision with an object are considered *infeasible*.

The *neighborhood* of a solution is simply defined by making a number of discretization steps to reachable regions from the current solution velocity tuple. The 4-neighborhood makes a step by either increasing or decreasing the current linear and angular velocity by one discretization step (Manhattan distance $= 1$). The 8-neighborhood takes in addition also the diagonal neighbors into account which are reachable in one discretization step (Moore neighborhood). 16-neighborhood are all neighbors reachable in two steps. This process continues until the whole search space is the neighborhood. Figure 22 visualizes the neighborhood structure.

Using *Local Search* (see Section 3.2.1) the neighborhood is either exhaustively searched for the best solution using Best-Improvement heuristic or stopped after finding the first improving solution using First-Improvement heuristic.

The following list includes the meta-heuristic algorithms used for the proposed approach.

TABU LIST

The *memory structures* (see Section 4) used in the proposed algorithms are very simple. Since the run-time requirements are very demanding, long term strategies are less suited to provide good results.

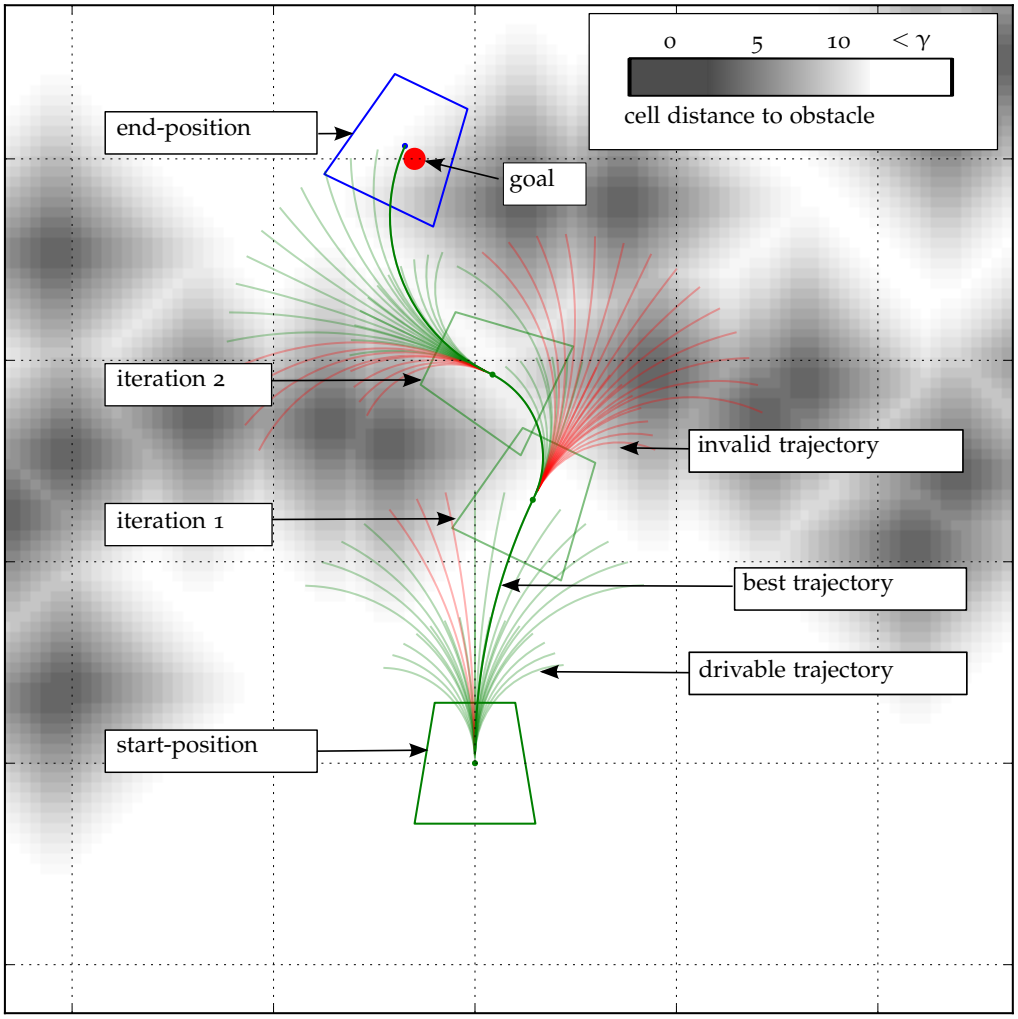


Figure 21: This figure shows three applications of the local planning step in a simulated experiment. The greyscale background image visualizes obstacle costs of a generated cost-map. The green trajectories are drive-able, whereas red trajectories collide with obstacles. At each local planning step all possible trajectories are weighted with respect to their distance to obstacles and their progression towards the goal destination. For example in the second step a valid trajectory which comes closer to the goal is rejected, while a shorter trajectory which stays farther away from obstacles is selected for execution. After performing three local planning applications the robot safely reaches the goal destination.(taken from [47])

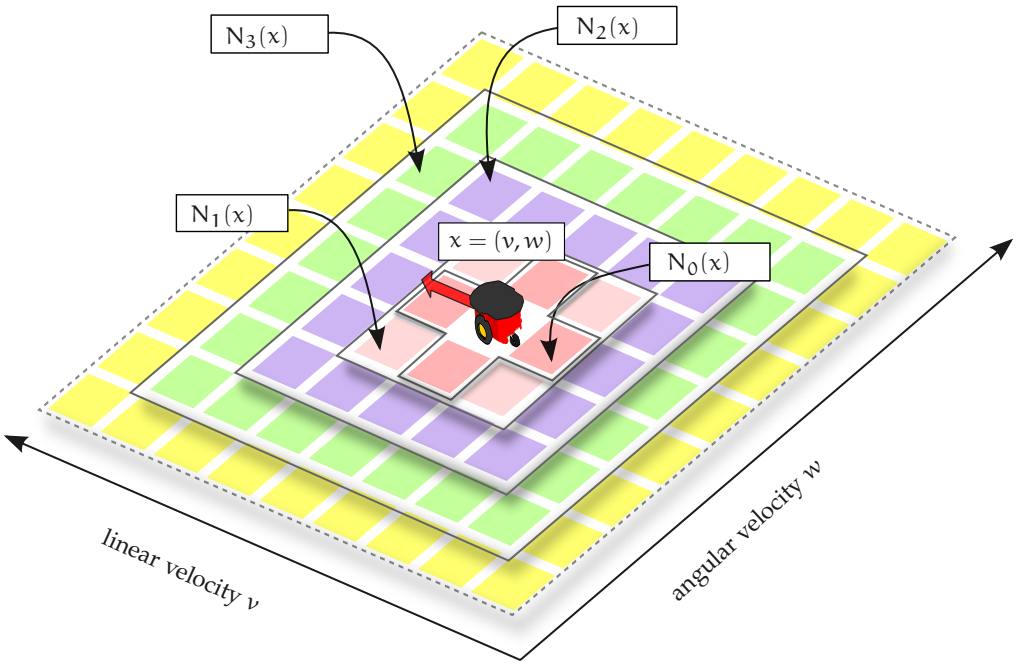


Figure 22: Neighborhood structure in the 2-dim velocity space.

The first strategy keeps all visited states during the search as tabu. They will not be considered as valid solution in future steps of the search process.

A slightly more sophisticated strategy is used in the extension of existing planner. Here two types of tabu lists are present.

1. long term memory: Keeps all *infeasible* velocity tuples which lead to collision. The are never considered again in future searches. This is possible since the solution space is not too large.
2. short term memory: Classical tabu list, with parametrized *tabu tenure* which keeps the most recently visited *feasible* velocity tuples.

The idea of avoiding the neighborhood of infeasible solutions and allowing to return to *good* feasible solutions during search was inspired by the grid like representation of the search space, which resembles the famous guessing game Battleship.

The goal of the game is to try to hit all of the opponent ships, which can be viewed as the dual problem to obstacle avoidance. If a player hits a ship during the game, a good strategy is to try out the immediate neighbors until the ship is destroyed. In obstacle avoidance the opposite may be more efficient. If an obstacle is encountered, avoid its immediate neighbors.

The tabu list is used by the other meta-heuristic algorithms during the Local Search.

ITERATED LOCAL SEARCH (ILS)

The main structure of this algorithm was already shown in Section 5. The algorithm searches iteratively a fixed neighborhood (eg. 4-, or 8-connected neighborhood) with Best Improvement heuristic for the Local Search.

Since the solution vector is only 2-dimensional, there is no need for complicated perturbation schemes. It is possible to either randomly generate a new linear velocity, a new angular velocity or both.

Using the *history* capability of ILS it is possible to apply the perturbation step after a fixed amount of iterations. Furthermore one can parametrize the algorithm to restart if the neighborhood is empty due to invalid neighbors or the application of the tabu list.

VARIABLE NEIGHBORHOOD SEARCH (VNS)

For the VNS algorithm local search is performed in one neighborhood until no improvement occurs. The used algorithm follows the scheme shown in Section 18.

The algorithm starts with either a random initial solution, or with the best solution found in the previous local planning step. In the *shaking* phase a new random solution in the current neighborhood is selected. If *shaking* does not yield a solution, because the whole neighborhood is tabu, or does not include a collision free trajectory, the next neighborhood is chosen.

An ordering of neighborhoods according to their size yields the neighborhood structures:

- $N_0(x) =$ 4-connected
- $N_1(x) =$ 8-connected
- ...
- $N_k(x) =$ k-steps reachable neighborhood.

The different neighborhoods are nested which yield the following structure of neighborhoods:

$N_0 \subseteq N_1 \subseteq \cdots \subseteq N_k$

In theory the union of all neighborhood is larger or equal the whole solution space of velocity tuples. Unfortunately large neighborhoods are very costly to evaluate. Therefore the neighborhood structure can be bounded above by a parameter k_{max}. If no improvement is made up to the $N_{k_{max}}$ neighborhood, a new initial solution is generated at random and the search starts up again.

Like the aforementioned ILS algorithm the VNS algorithm can use the same *memory structures* during the search.

The VNS algorithm can be used with Best-, or First-Improvement heuristic for the Local Search.

4 EVALUATION

To evaluate the proposed method two classes of experiments are carried out. In a first step an example planner is implemented which mimics the local planning step and provides an easy way to test the trajectory evaluation separated from the rest of a conventional planning system.

With the gained data the algorithms are improved and used to extend existing local planner. A fully simulated environment and robotic model using a sophisticated physic engine together with a complete robot navigation system is used to compare the original implementation of the local planner with the meta-heuristic approach.

4.1 EXPERIMENTS WITH SAMPLE PLANNER

The trajectory sampling and selection of a common DWA approach are implemented in python minimizing a simpler cost function $f_c(v, w)$ (cf. Equation 4), where $f_g(v, w)$ is the distance of the center of the robot in the end position to a predefined goal position, and $f_o(v, w)$ is the maximal distance to an obstacle on the trajectory path.

$$f_c(v, w) = \alpha f_g(v, w) - \beta f_o(v, w) \tag{4}$$

To select a benchmark cost a brute force search is performed on random generated test instances, evaluating a fixed number of trajectories. The time the algorithm needs to find this benchmark solution is used to compare their performance.

All algorithm are tested using different minimal, and maximal velocities to account for different acceleration limits.

The weighting coefficients of the cost function are fixed in our case to $\alpha = 0.01$ and $\beta = 1$. The local goal is also at a fixed location in the map. The step size of the collision test is fixed to 0.015 meter. Forward simulation time is fixed to one second.

The following 60 test instances include different obstacle counts and random placement of quadratic obstacles:

- 15 instances with 1 obstacle and side length 1 meter.
- 15 instances with 3 obstacles and side length 1 meter.
- 15 instances with 5 obstacles and side length 0.5 meter.

- 15 instances with 25 obstacles and side length 0.1 meter.

The instances simulate a snapshot of the local environment of the robot at a given time point, which is used as a local map for input of the local planner. The resolution of the maps is fixed to 0.05 meter/pixel, resulting in a quadratic map of size 7.5 meter. Figure 23 illustrates random instances with differ in size and number of obstacles.

The following list shows the tested algorithm:

- **Random Search with Tabu List:** A repeated random guess of a velocity tuple (v, w) (Random).
- **Iterated Local Search:** Performing Iterated Local Search with 4, 8 ,and 16 neighbors and Tabu List (ILS4, ILS8, ILS16).
- **Variable Neighborhood Search:** Variable Neighborhood search with Best- ,and First-Improvement heuristic, and Tabu List (VNSB, VNSF).

4.2 EXPERIMENTS EXTENDING EXISTING LOCAL PLANNER

To extend an existing local planner the navigation system implemented in the ROS framework is used, which provides a variety of different planning methods. The planning system chosen for testing comes ready to use in the implementation of the navigation stack of the ROS framework, which was introduced and implemented by Marder-Eppstein (see [32]). The local planning system consists of two planners based on *DWA* implementation and *Trajectory Roll-out* (see Section 2.5).

To simulate the environment and the robot the simulation framework Gazebo is used, which provides a robust physical engine and is one of the most popular simulation engines in the field of mobile robotics.

In Figure 24 the 3D-model, which is a simple artificial building with four corridors, used for the simulation together with a snapshot of the planning information is depicted.

In Figure 25 the second 3D-model used for testing is shown. Based on a real map this office environment includes more difficult planning situations, like small doorways and corridors.

To compare the meta-heuristic approach with the unaltered versions of the ROS-planners, a simulated robot has to fulfill a simple navigation task within the provided environments. A route is provided by marker points which have to be approached one by one until a round trip is completed. To account for the probability aspect of the meta-heuristic implementations several round trips have to be accomplished until the experiments stops.

At each call of the local planner first the original brute force evaluation of the trajectories is conducted and the best score together with the time needed to find the best trajectory is recorded. Immediately after a solution is found,

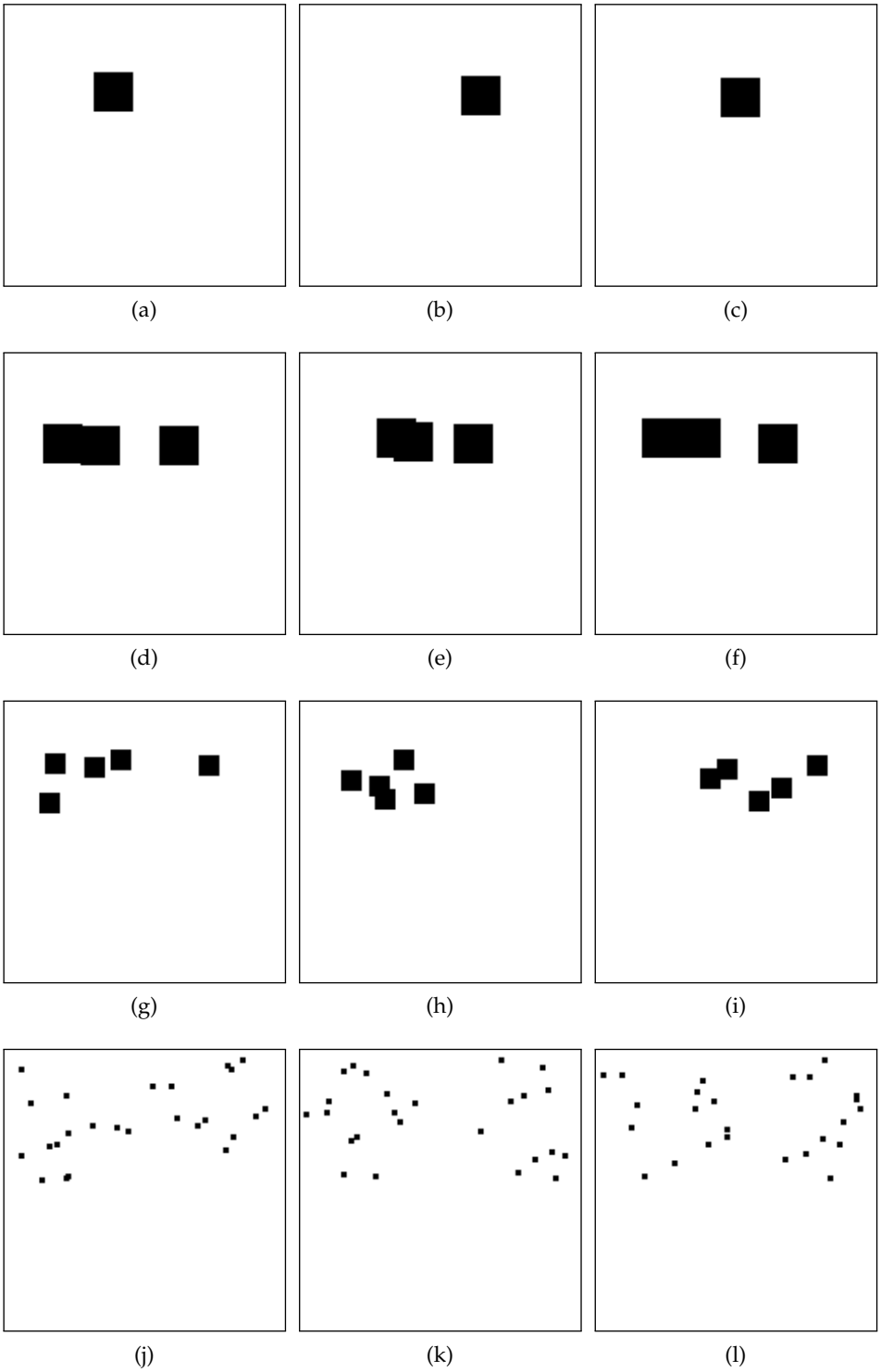
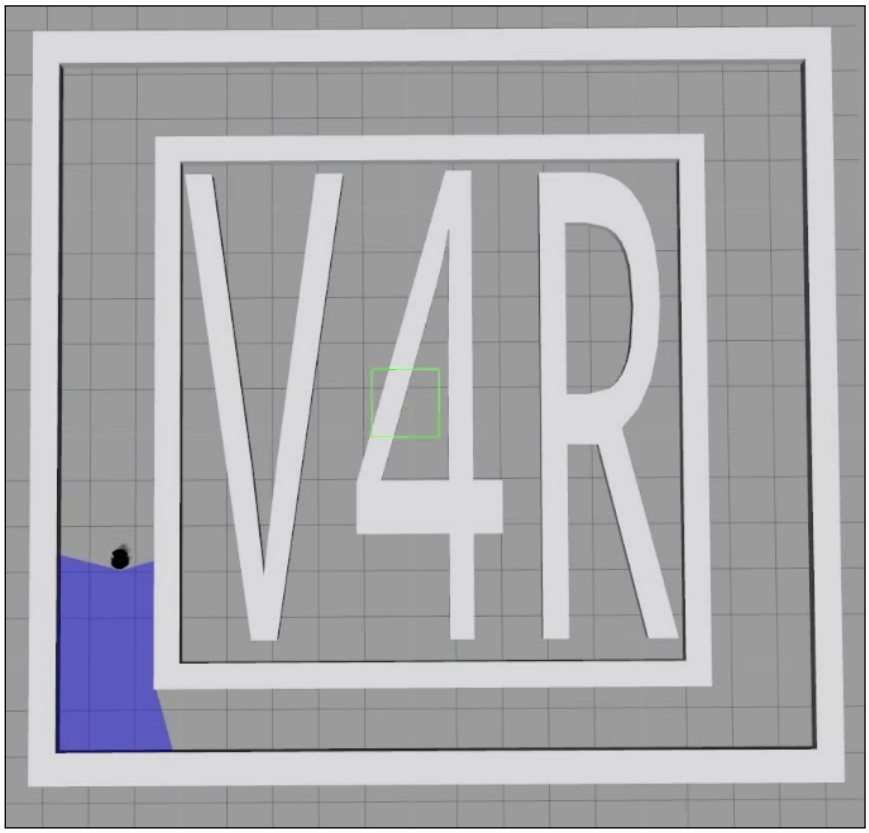


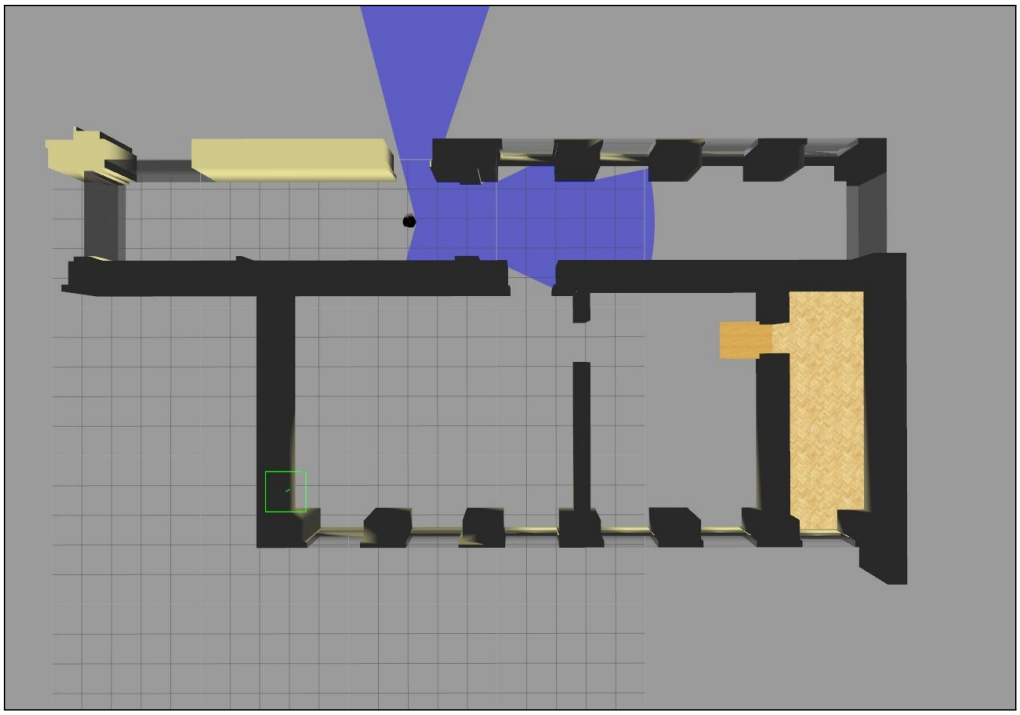
Figure 23: Figures (a)-(l) show randomly generated local obstacle maps. The instances differ in number and size of obstacles and are used for local costmap creation. Up to 60 instances are used to evaluate the proposed method.



(a)

(b)

Figure 24: Figure (a) shows the 3D model of an artificial building used for the experiments with the simulation software Gazebo. Figure (b) depicts planning information used during the experiments.



(a)

(b)

Figure 25: Figure (a) shows the 3D model of an office environment used for the experiments with the simulation software Gazebo. Figure (b) depicts planning information used during the experiments.

the VNS search algorithm searches for the same exact best score. Again the performance time until completion of the algorithm is recorded. The performance time of the original and the altered version of the local planner are used for comparing the different algorithms.

The following list shows the tested algorithm:

- **VNS-ROL:** This implementation uses Variable Neighborhood Search with Best Improvement together with the Trajectory Roll-out planner.
- **VNS-DWA:** This implementation uses Variable Neighborhood Search with Best Improvement together with the Dynamic Window Approach planner.

4.3 RESULTS

All experiments account for the randomness of the proposed methods by running the algorithm multiple times.

The results are graphically visualized using box plots. The colored box is bounded by the lower quartile and the upper quartile of the data (25% and 75% of the data). The median is indicated by a straight line through the box. The upper whisker extending at one end of the box indicate the last data point which lies below the 75% quartile plus 1.5 times the inter quartile range (IQR). The lower whisker on the other end indicates the last data point which lies above the 25% quartile minus 1.5 times the IQR. Points outside of the whisker range are considered outliers. As a rule of thumb if the boxes between two measured data sets do not overlap, the difference between those data sets is significant.

Figure 26 shows a commmon boxplot.

4.3.1 *Sample planner*

In this set of experiments the algorithms were tested using the sample planner and the artificial sensor map instances. These tests were performed on a 2.4 GHz, Intel Core 2 Duo processor using 4 GB RAM.

4.3.1.1 *Influence of trajectory size*

The algorithms were applied to all 60 instances to evaluate a broad spectrum of possible environments. The main reason conducting this experiment is to get a first impression on the usefulness of applying meta-heuristics in the context of local planning. Figure 27 illustrates the results using 240 trajectories, and using 2400 trajectory samples.

The results show that all algorithms, including RST, outperform the Brute Force generate and test method significantly. As expected increasing the num-

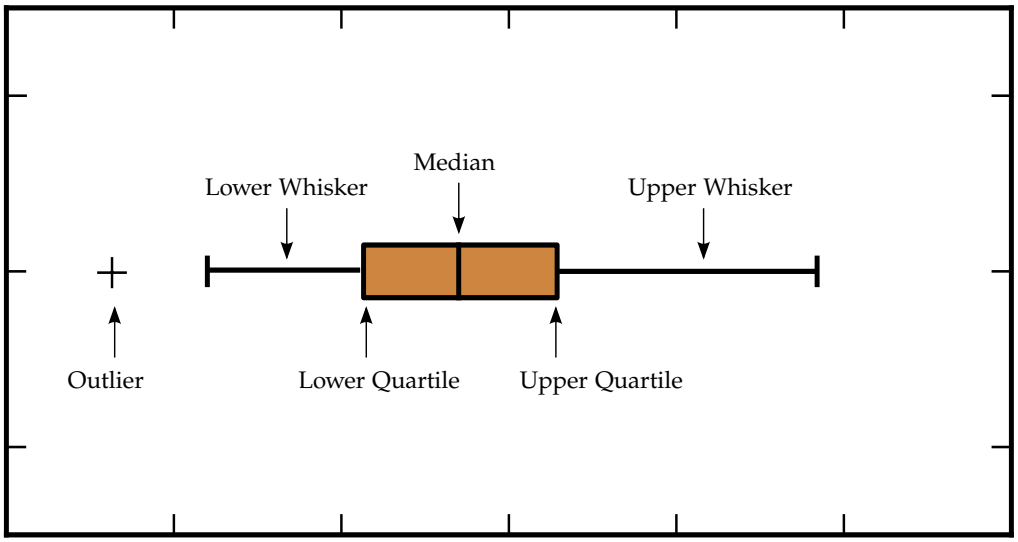


Figure 26: A classical boxplot diagram.

ber of trajectories greatly favors the meta-heuristic algorithms, since they benefit from larger search spaces.

Notice that ILS and VNS algorithms differ apparently from the RST by exhibiting much smaller variance in their test results, indicating that randomization alone is not enough to achieve very good and stable performance. Furthermore the VNS exhibit a more stable performance than the ILS methods. Comparing the ILS algorithms reveals the connection of the search space size to the size of the neighborhood. A small number of trajectories benefits smaller sized neighborhoods, whereas increasing the number of trajectories benefits larger neighborhoods.

4.3.1.2 *Influence of trajectory size by specific instances*

The following tests include the VNSF, VNSB and only one ILS4 algorithm for comparison. The algorithms are executed with specific world instances, and repeated 50 times. The results in Figure 28 show the application of the algorithm using 240 trajectories. Again all tested algorithms significantly outperform the Brute Force method.

Increasing the number of trajectories favors the VNS algorithms over the ILS4 algorithm. The results in Figure 29 show the application of the algorithm using 960 trajectories.

In contrast to the smaller number of trajectories, the results of the ILS4 algorithms shows that a too small environment will quickly degrade to random search if the number of trajectories increases. Here the use of a neighborhood structure pays off and the VNS approaches perform evidently better than ILS. In addition, the results show that the algorithms perform good independent of number and size of obstacles.

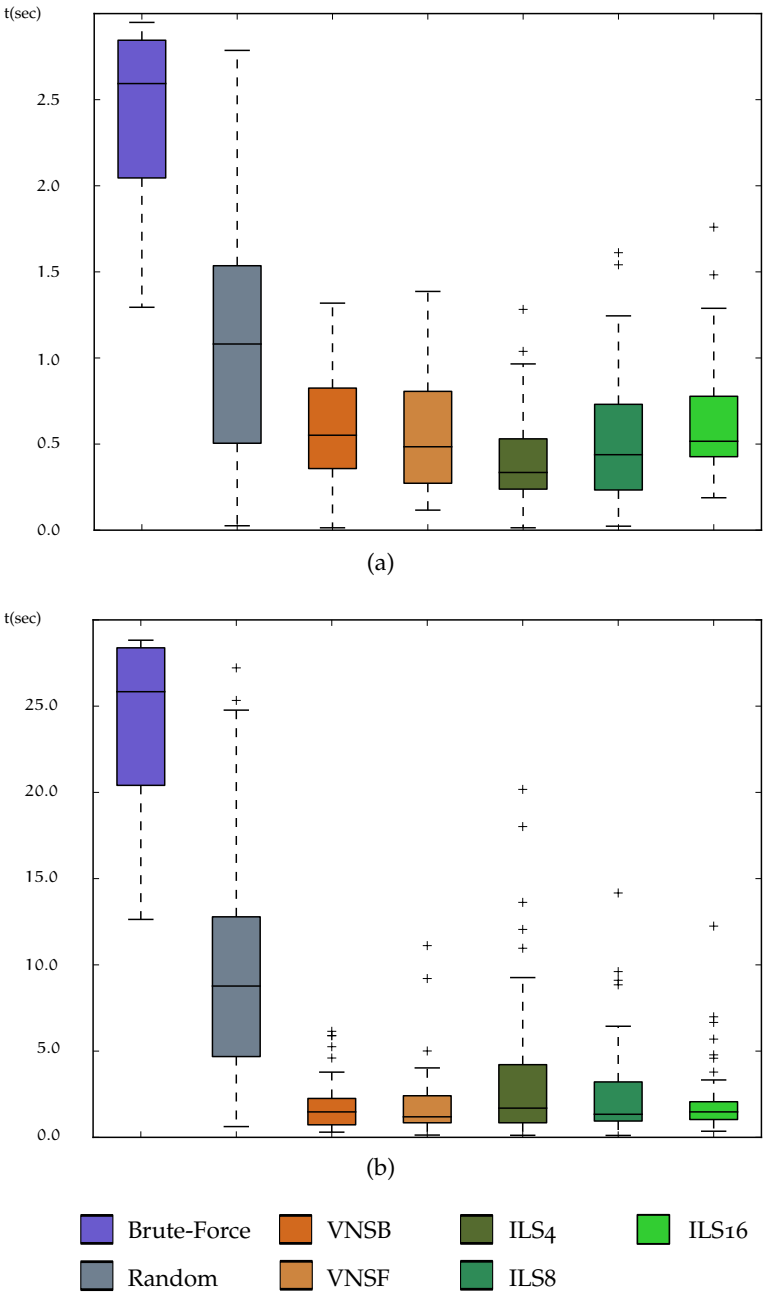


Figure 27: This figure shows the results of testing all 60 randomly generated instances. The top figure shows the run time performance for 240 trajectories, and the bottom figure for 2400 trajectories. Compared to brute force search, the meta-heuristic algorithms show a significant improvement. (adapted from [47])

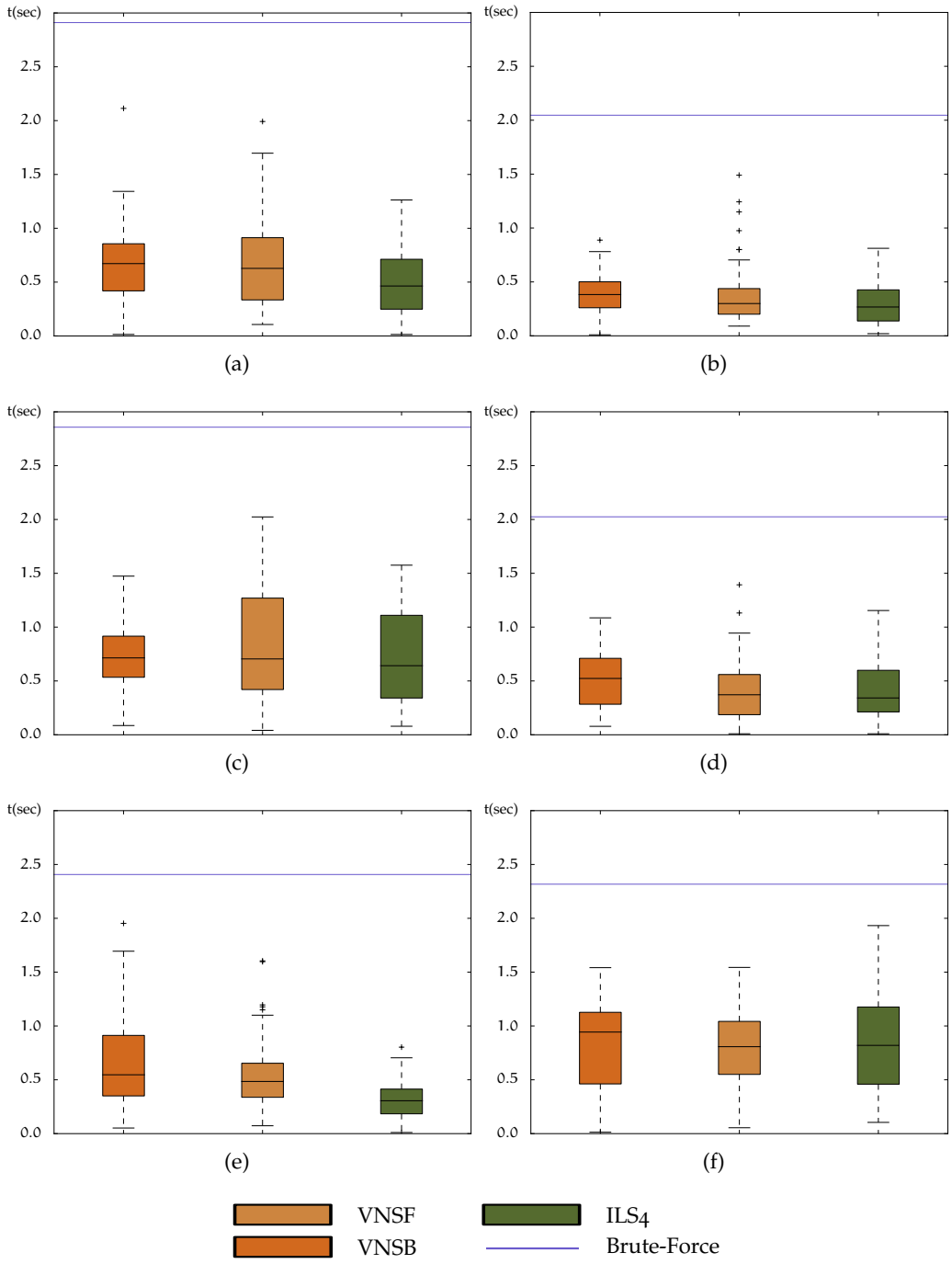


Figure 28: The results of 50 consecutively executions with 240 trajectories. Plots (a,b) shows the result for 1 and 3 obstacles with size 1 meter. Plots (c,d) shows the result for 5 obstacles with size 0.5 meter, and plots (e,f) shows the result for 25 obstacles with size 0.1 meter. The blue line marks the run time for brute force search, which is used as a benchmark.(adapted from [47])

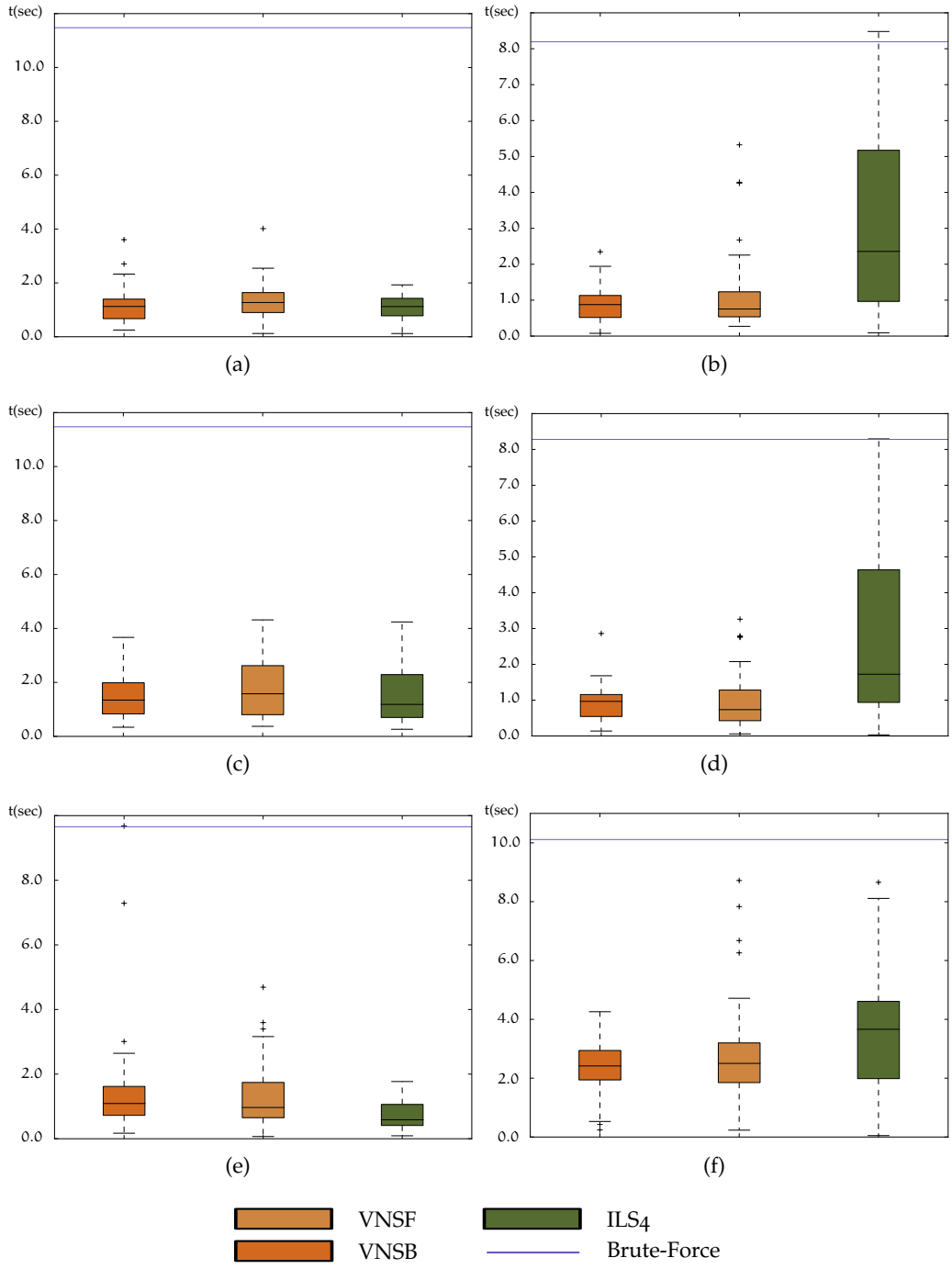


Figure 29: The results of 50 consecutively executions with 960 trajectories. Plots (a,b) shows the result for 1 and 3 obstacles with size 1 meter. Plots (c,d) shows the result for 5 obstacles with size 0.5 meter, and plots (e,f) shows the result for 25 obstacles with size 0.1 meter. The blue line marks the run time for brute force search, which is used as a benchmark.(adapted from [47])

4.3.1.3 *Influence of velocity bounds*

The next analysis focuses on the influence of the velocity bounds on the performance of the local planning step. The trajectory size is fixed to 240 trajectories and the velocity bounds are varied. Since the previous test has shown that the VNS approaches are more promising, the following experiments omit the ILS algorithms. 5 experiments with 50 runs and increasing acceleration windows are conducted. The test are again performed on different sizes and numbers of obstacles.

In Figure 30 the results on instances with 1 to 3 obstacle with size 1 meter are presented.

Figure 31 presents the results on instances with 5 obstacles with size 0.5 meter.

In Figure 32 the results on instances with 25 obstacles with size 0.1 meter are presented.

All meta-heuristic algorithms outperform the Brute Force generate and test method on all tested instances, regardless of the differences in the velocity bounds. It can be observed that the small instances with many obstacles provide the most difficulties for the proposed methods. This is increased by using a larger velocity space. While increasing the velocity windows clearly has an obvious negative effect on the performance of both the brute force and the meta-heuristic methods, the meta-heuristic algorithms soften this influence to a large degree.

4.3.2 *Existing local planner*

This set of experiments are conducted using the simulation software Gazebo for the artificial and office environment. All of these tests are performed on an Intel Core2 Quad CPU with 2.66GHz and 8 GB RAM.

4.3.2.1 *VNS and Trajectory Roll-out planner*

The Trajectory Roll-out planner variant of the ROS-navigation stack implementation was altered to use the VNS approach (VNS-ROL) and compared to the unaltered version in all of the following results.

The first experiment was executed using only one tabu list which holds all visited solutions. The robot was tested with 300 trajectories in the logo environment and had to perform 15 roundtrips. The results in figure 33a show that the VNS-ROL algorithm significantly outperforms the original approach.

In a second experiment the experiment time was extended to 80 roundtrips in the logo environment. Again the local planner used 300 trajectories. This time both memory structures were used with a tabu tenure of 7. The results in figure 33b show again that the VNS-ROL algorithm significantly outperforms the original approach.

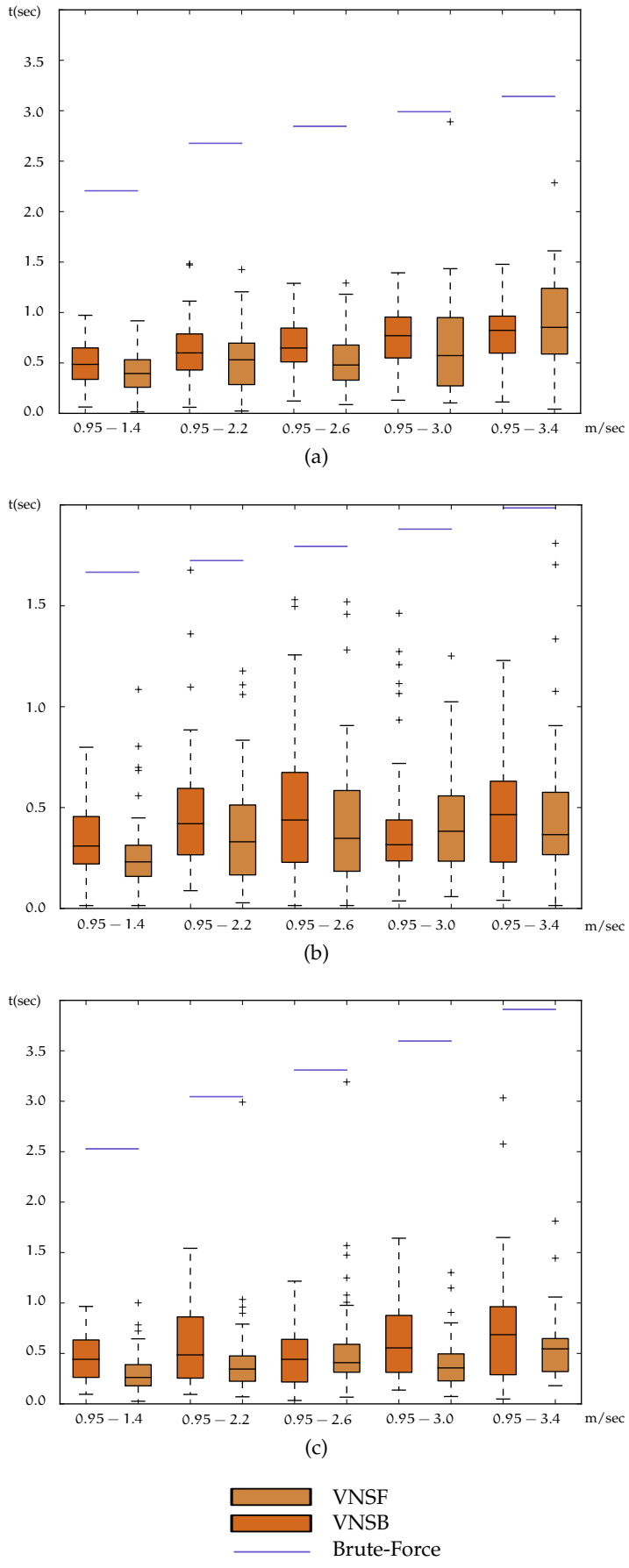


Figure 30: The results of 50 consecutively executions with 240 trajectories, on instances with 1 (a) and 3 obstacles (b-c) with size 1 meter. The blue line marks the run time for brute force search, which is used as a benchmark.

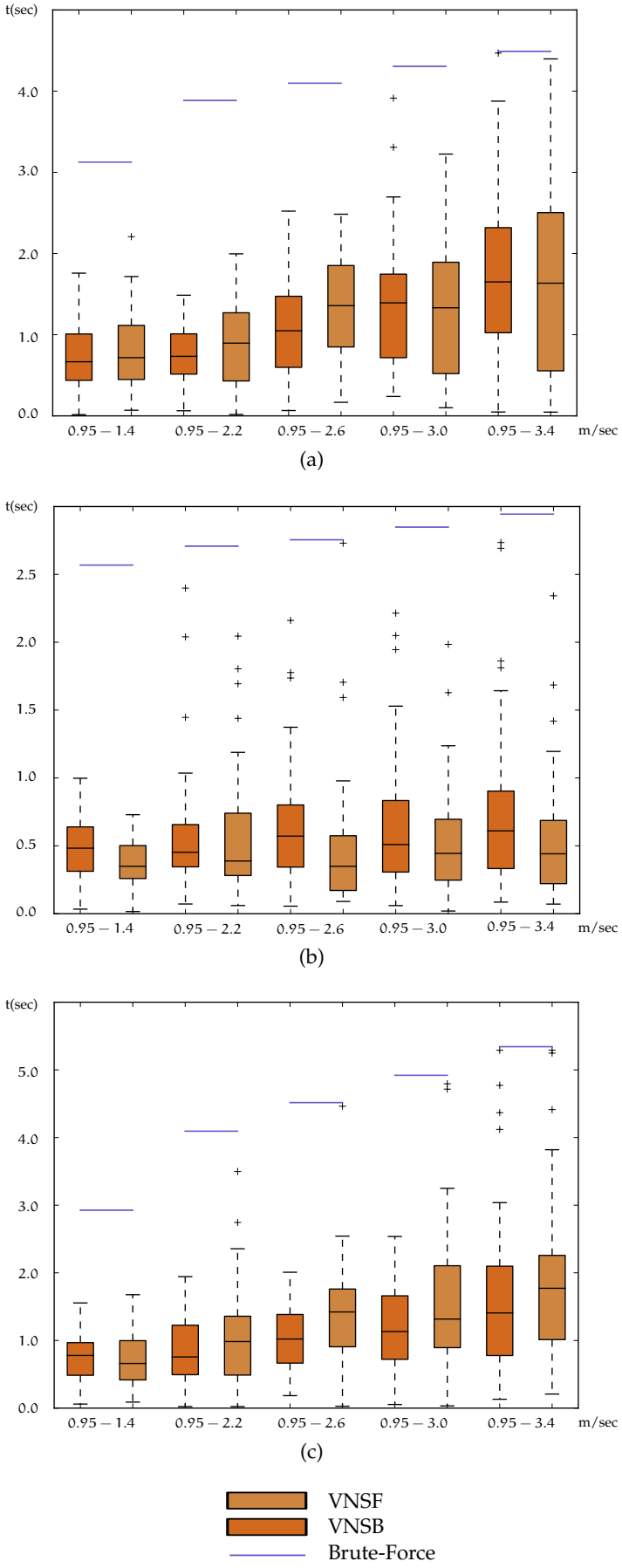


Figure 31: The results of 50 consecutively executions with 240 trajectories, on instances with 5 obstacles with size 0.5 meter. The blue line marks the run time for brute force search, which is used as a benchmark.

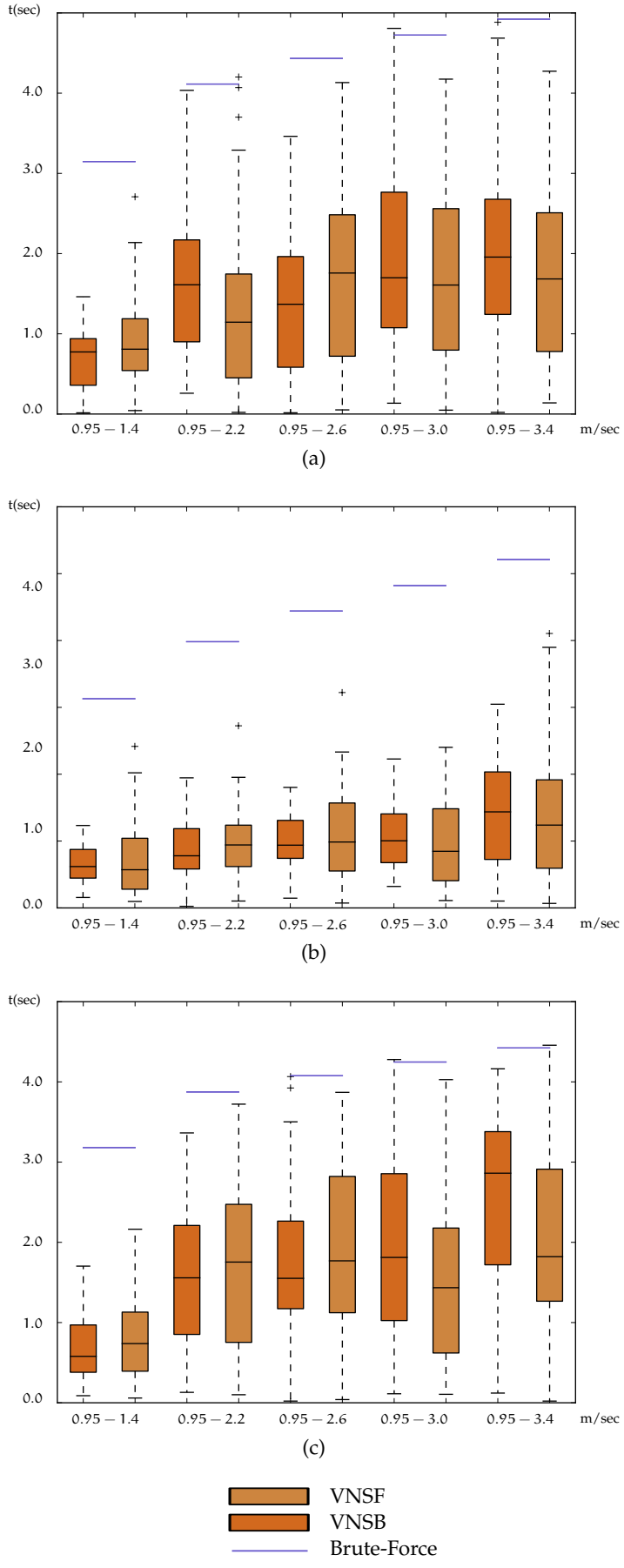
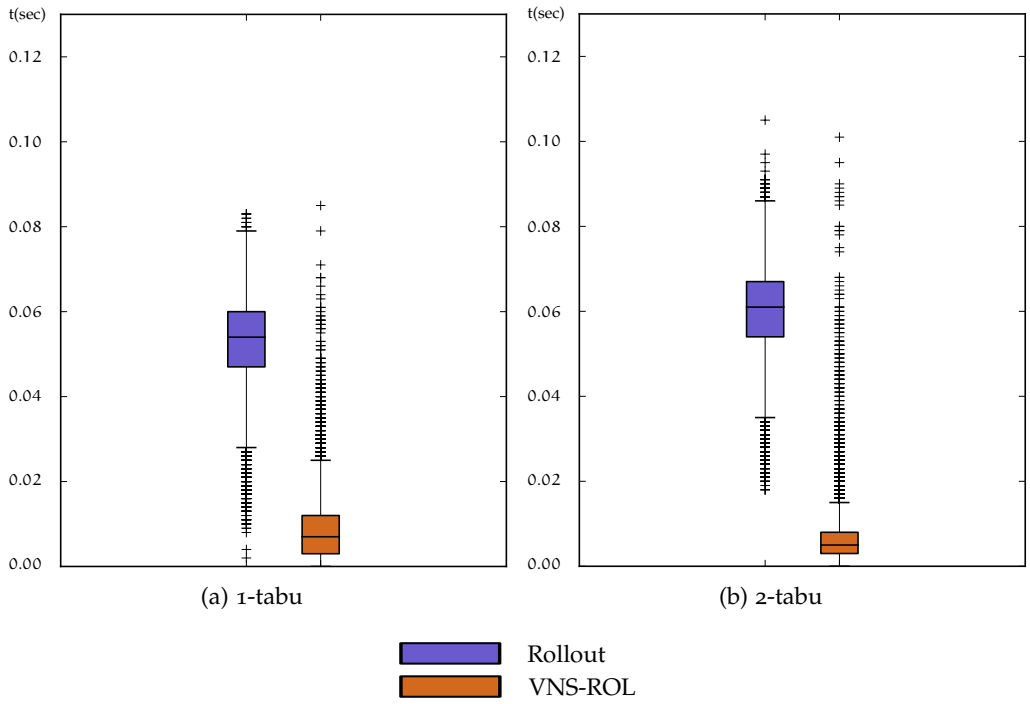


Figure 32: The results of 50 consecutively executions with 240 trajectories, on instances with 25 obstacles with size 0.1 meter. The blue line marks the run time for brute force search, which is used as a benchmark.



(a) 1-tabu (b) 2-tabu

Figure 33: These figures show the result of using the VNS-ROL algorithm in the logo environment. The results of 15 roundtrips with one tabu list is shown in (a). Figure (b) shows the result of 80 roundtrips using an additional tabu list with tenure 7.

Using the short term memory structure did have a small but not significant performance increase, although less outliers and slightly smaller variance of the measurements is observable.

The next four experiments show the effects of increasing the number of trajectory samples and the influence of the tabu tenure. This time the robot had to travel 40 roundtrips in the office environment. The planners are tested with 300 and 600 trajectories with tabu tenure set to 10 and 100. The results are shown in figure 34.

As in the experiments with the sample planner, the larger trajectory size increased the performance difference between the original brute-force method and the VNS approach. Increasing the size of the tabu tenure did have a negative effect on the performance, showing that a small tenure is not only enough to avoid cycling but is also faster due to the smaller lookup time in a shorter list.

4.3.2.2 *VNS and DWA planner*

This section contains the results of the experiments with the DWA variant of the ROS local planner and the VNS approach (VNS-DWA).

Starting with 15 roundtrips using 300 trajectories in the logo environment the meta-heuristic algorithm shows similar performance as the VNS-ROL algorithm.

The long run experiment with 80 roundtrips revealed a problem for the VNS-DWA approach with 2 tabu lists, which was not detected during the tests with the Rollout planner. In some rare cases the meta-heuristic approach did use exceptional long time to find the best solution, leading to instability of the system. Even extending the tabu tenure to half of the solution size did not solve this problem.

Through further analysis the problem was identified to occur only when the robot approached a milestone in the roundtrip with very small velocity. Since the robot did barely move at all the corresponding cost function had the same value for most velocity tuples, providing no usable gradient for guiding the local search.

In order to avoid this situation the VNS-DWA algorithm was altered to use the brute force method whenever the velocity of the robot drops beyond a predefined very small threshold value.

The result of these experiments are shown in figure 35

For the next experiments the robot had to travel 40 roundtrips in the office environment. The planners are tested with 300 and 600 trajectories with tabu tenure set to 10 and 100. The results are shown in figure 36.

The VNS-DWA algorithm outperformed the original approach in all experiments. Again the experiments with the larger trajectory samples favor the meta-heuristic approach. In contrast to the results of the VNS-ROL experiments, the larger tabu tenure did not show to have any noticeable effect.

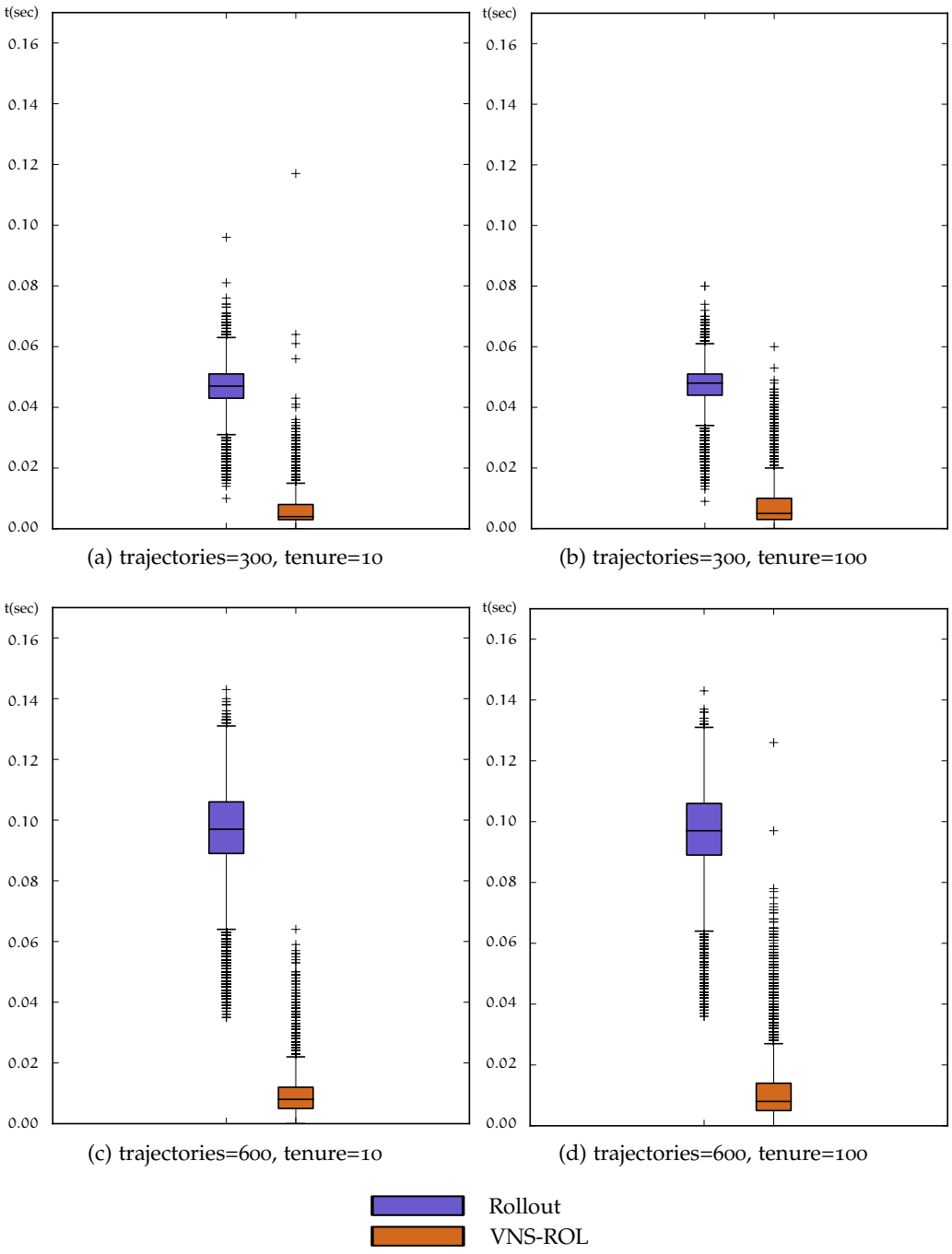


Figure 34: These figures show the result of using the VNS-ROL algorithm in the office environment. In each experiment the robot has to drive 40 roundtrips in the building. Figure (a) shows the result using 300 trajectory samples with tabu tenure 10 while Figure (b) uses a tenure of 100. Figure (c) shows the result of 600 trajectories with tabu tenure 10 and Figure (d) a tenure of 100.

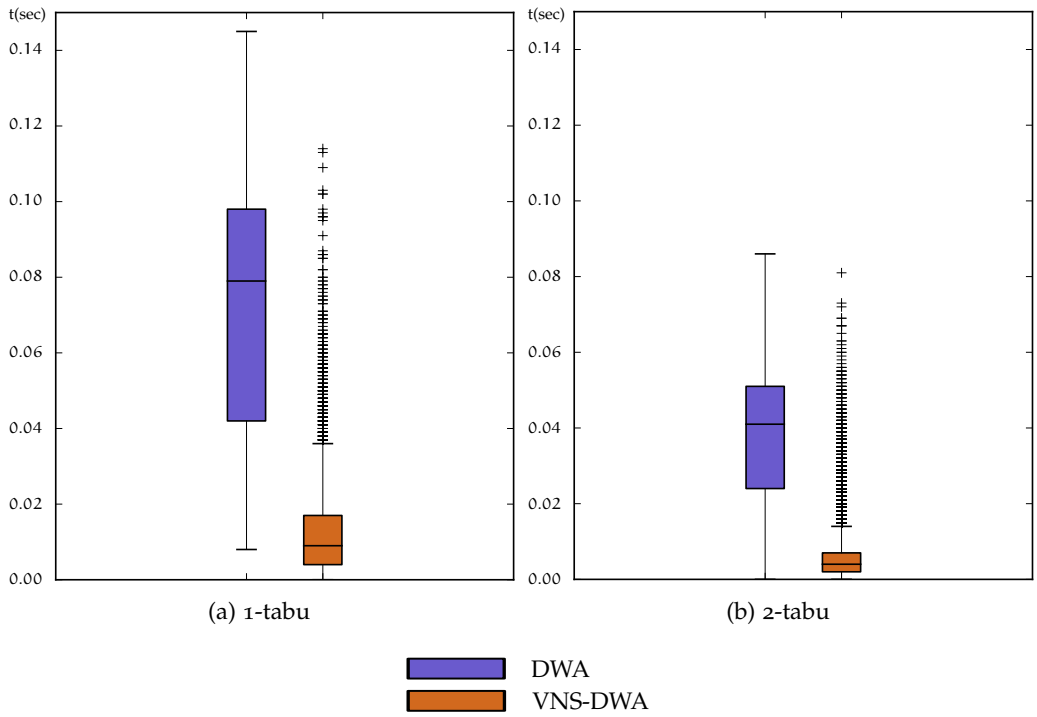


Figure 35: These figures show the result of using the VNS-DWA algorithm in the logo environment. The results of 15 roundtrips with one tabu list is shown in (a). Figure (b) shows the result of 80 roundtrips using an additional tabu list with tenure 7.

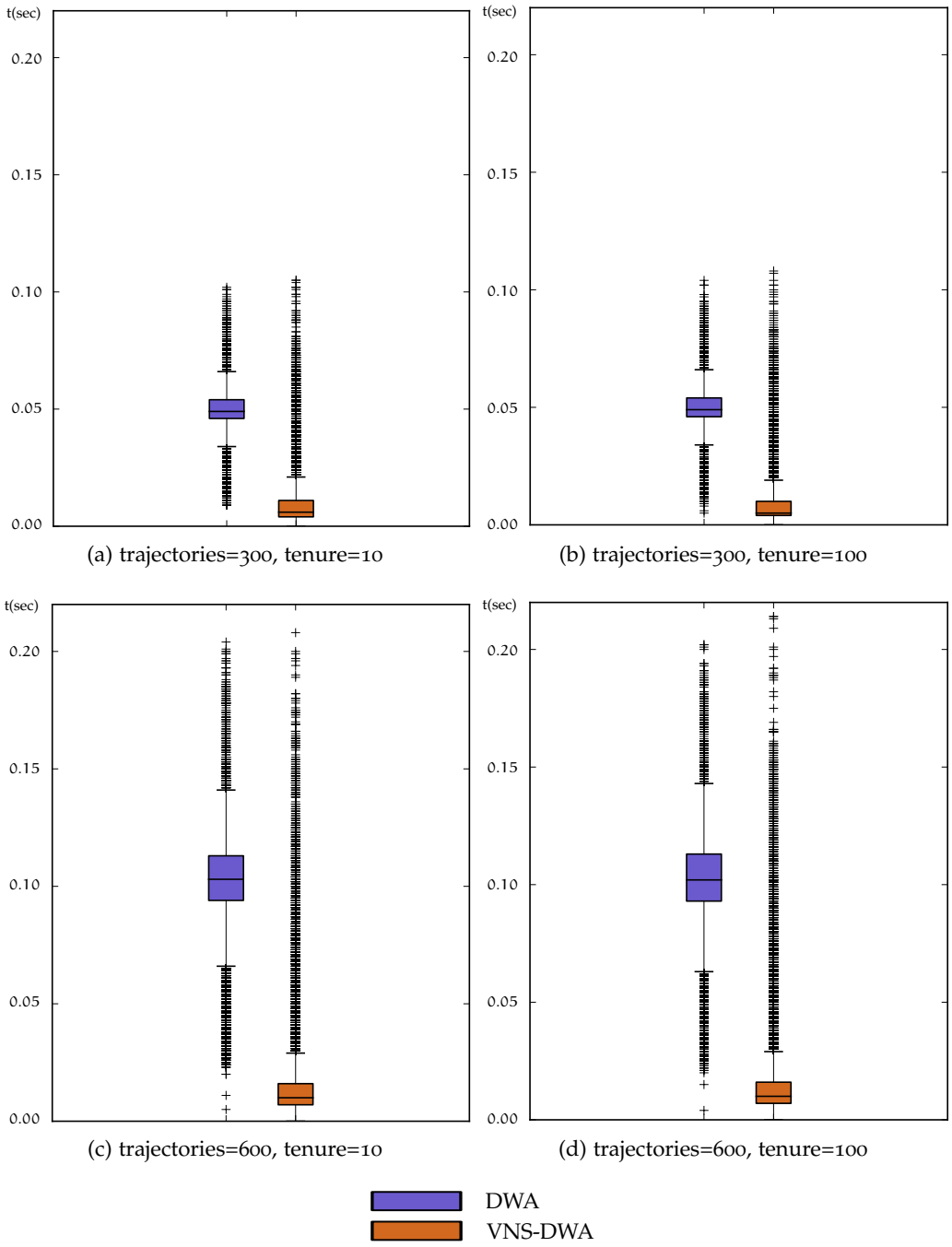


Figure 36: These figures show the result of using the VNS-DWA algorithm in the office environment. In each experiment the robot has to drive 40 roundtrips in the building. Figure (a) shows the result using 300 trajectory samples with tabu tenure 10 while Figure (b) uses a tenure of 100. Figure (c) shows the result of 600 trajectories with tabu tenure 10 and Figure (d) a tenure of 100.

4.4 SUMMARY

As it is the case for nearly all optimization problems, the No Free Lunch theorems [53] apply also to the local planning domain. So it is no surprise that looking at all the results of the sample planner, there is no clear winner among the algorithms, but Variable Neighborhood search with tabu list and Best, or First Improvement heuristic seem to yields the best and most stable overall performance.

In general the run time of the python implementation used for the example planner is not very efficient compared to tuned C++ implementations. Therefore the absolute numbers of the run time evaluations should be handled with care.

After analysis of the results with the sample python planner, the VNS approach is selected to be integrated into a navigation system based on a C++ implementation in the ROS framework.

The tests conducted with a simulation software show that the previous results carry over to a real navigation scenario. Extensions to the Trajectory Rollout planner within the navigation stack of ROS using the VNS algorithm outperform the unaltered version significantly.

While the application of the VNS search for the Roll-out planner was straight forward, the tests revealed a problem with this approach for the DWA planner. For very low velocities the corresponding costfunction did not provide enough information for a fast local search, leading to instability of the system. In order to deal with these situations, the VNS-DWA algorithm falls back to brute force evaluation of trajectories whenever the robots moves very slow.

After this small change the VNS-DWA could be successfully tested and the results showed, that the performance increase was still significant.

4.5 IMPLEMENTATION DETAILS

The example planner was implemented using *python* version 2.7 (available at `http://python.com`). This allows for a fast prototype implementation of a local planner without the overhead and interruption of other parts of full planning systems. Hence the optimization algorithms could be evaluated focusing on the specific trajectory selection part.

For visualization, geometry related processing, and the creation of test scripts the python packages *Numpy* (available at `http://numpy.org`), *Python Imaging Library* (available at `http://pythonware.com`) and *Matplotlib* (available at `http://matplotlib.org`) were an essential part of the created software.

The VNS algorithm used in the navigation framework within ROS, was created using C++ and the BOOST-C++ Libraries (`http://boost.org`).

The source code was compiled using *GCC* 4.6.4 with optimization enabled.

Without the use of these exceptional open-source libraries the work presented in this thesis would not have been possible.

5 CONCLUSIONS

In this work a promising approach to improve performance of existing local planning systems which are based on generate and test trajectory methods by using meta-heuristic search strategies was analyzed.

To this end this thesis presented a thorough overview of local planning and obstacle avoidance methods. Identifying trajectory selection as the main part of local planning tasks for improvement, well known meta-heuristic search algorithms were introduced for optimizing the selection process.

From the large family of meta-heuristic algorithms the following search strategies were selected to substitute the brute force approach of trajectory selection in local planning methods based on *trajectory roll-out* and *DWA*:

- The basic algorithmic structure of the single solution based meta-heuristics *Iterated Local Search (ILS)* and *Variable Neighborhood Search (VNS)*.
- An appropriate formulation of a set of *neighborhood* structures, which are used during *Local Search* with *best improvement* and *first improvement* heuristic.
- Two basic *memory structures* inspired from *tabu search* to avoid infeasible solutions and short cycles during the search process.

To evaluate the proposed method a set of random generated environments simulating sensor information for one planning step were created. These instances were used to test the algorithms using an example planner. This allowed a focused investigation and improvement of the trajectory selection part separated from the overhead of full planning systems.

The results of these experiments with the example planner showed, that the meta-heuristic algorithms provide significant performance improvement on all of the test instances. Especially the VNS implementations provided very good and stable results.

In a next step the VNS approach was used to extent a popular existing planner implementation and tested using a sophisticated simulation environment with realistic scenarios and physics. Two configurations were tested in an virtual building:

- VNS-ROL: Using VNS with best improvement heuristic to extend the roll-out method of the planner with continues acceleration limits.
- VNS-DWA: Using VNS with best improvement heuristic to extend the classical DWA method of the planner.

Both extensions showed superior run time performance compared to their unaltered counterparts.

Summarizing the meta-heuristic algorithms are able to increase the performance of local planning systems based on trajectory generate and test methods. This allows the local planner to improve on *reactivity*, the used *resolution* for collision tests, the *simulation-time* for look-ahead and the number of *trajectories* used to model the robots motion capabilities.

5.1 FURTHER RESEARCH

Applying meta-heuristic search to trajectory selection of local planners like DWA shows to be a promising step in using the power of these search procedures in the context of local planning. Therefore the following directions for further research are suggested:

INVESTIGATE ADDITIONAL META-HEURISTICS

This work presented the use of a very limited selection of single solution based meta-heuristics. Therefore it would be of interest to investigate related algorithm like GRASP [11], reduced VNS, or Simulated Annealing [23] or even population based meta-heuristics. In addition, developing and analyzing more sophisticated neighborhood structures would be recommended.

APPLICATION TO HIGH DOF ROBOTIC MODELS

The proposed method might also be applicable for robot models of higher degree of freedom, since dealing with large trajectory samples is a particular strength of meta-heuristic search.

EXTENDING TO THREE-DIMENSIONAL SPACE

This work considers exclusively two-dimensional workspaces. Application of this method to three-dimensional spaces is an interesting challenge and would allow the use of new obstacle avoidance methods for robotic models like unmanned air vehicles (UAV).

EXTENDING OTHER PLANNING SYSTEMS

The applicability to similar path-planning methods using trajectory samples, like Curvature Velocity Method [46], would be of interest to show the general applicability of this method.

INCREASING THE PLANNING HORIZON

The extended planning methods simulate trajectories by applying a constant amount of velocity over a given time. This restriction simplifies the planning step significantly and allows for brute force evaluation of trajectories. With the help of the proposed method the evaluation of chaining short series of different velocity commands may still be feasible and

would allow for a better lookahead simulation, allowing more sophisticated maneuver.

BIBLIOGRAPHY

[1] Mauro Birattari, Luis Paquete, Thomas Strutzle, and Klaus Varrentrapp. Classification of metaheuristics and design of experiments for the analysis of components tech. rep. aida-01-05. 2001.

[2] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

[3] Johann Borenstein and Yoram Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, 1991.

[4] Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965.

[5] N Buniyamin, N Sariff, WAJ Wan Ngah, and Z Mohamad. Robot global path planning overview and a variation of ant colony system algorithm. *International journal of mathematics and computers in simulation*, 5(1):9–16, 2011.

[6] N Buniyamin, WAJ Wan Ngah, N Sariff, and Z Mohamad. A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, 5(2):151–159, 2011.

[7] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[8] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[9] Joseph W Durham and Francesco Bullo. Smooth nearness-diagram navigation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 690–695. IEEE, 2008.

[10] Joseph F. Engelberger. Health-care robotics goes commercial: the helpmate experience. *Robotica*, 11:517–523, 11 1993. ISSN 1469-8668. doi: 10.1017/S0263574700019354. URL http://journals.cambridge.org/article_S0263574700019354.

[11] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[12] David Fischinger, Peter Einramhof, Walter Wohlkinger, K.Papoutsakis, P. Mayer, P. Panek, T. Koertner, S. Hoffmann, A. Argyros, Markus Vincze, Astrid Weiss, and C. Gisinger. Hobbit - the mutual care robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, 2013.

[13] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, Mar 1997. ISSN 1070-9932. doi: 10.1109/100.580977.

[14] Michel Gendreau. An introduction to tabu search. In Fred Glover and GaryA. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 37–54. Springer US, 2003. ISBN 978-1-4020-7263-5. doi: 10.1007/0-306-48056-5_2. URL http://dx.doi.org/10.1007/0-306-48056-5_2.

[15] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition, 2010. ISBN 1441916636, 9781441916631.

[16] Brian P. Gerkey and Kurt Konolige. Planning and control in unstructured terrain. In *In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, 2008.

[17] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, May 1986. ISSN 0305-0548. doi: 10.1016/0305-0548(86)90048-1. URL http://dx.doi.org/10.1016/0305-0548(86)90048-1.

[18] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. ISBN 079239965X.

[19] Erico Guizzo. Three engineers, hundreds of robots, one warehouse. *Spectrum, IEEE*, 45(7):26–34, 2008.

[20] Erico Guizzo. How googles self-driving car works. *IEEE Spectrum Online, October*, 18, 2011.

[21] Pierre Hansen, Nenad Mladenović, and José A Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[22] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968.

[23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[24] Domokos Kiss and Gábor Tevesz. Advanced dynamic window based navigation approach using model predictive control. In *Methods and Models in Automation and Robotics (MMAR), 2012 17th International Conference on*, pages 148–153. IEEE, 2012.

[25] Nak Yong Ko and Reid G Simmons. The lane-curvature method for local obstacle avoidance. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 3, pages 1615–1621. IEEE, 1998.

[26] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Oskar von Stryk, and Uwe Klingauf. Robocuprescue 2014 - robot league team hector darmstadt (germany). Technical report, Technische Universität Darmstadt, 2014.

[27] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. ISBN 0521862051. Available at http://planning.cs.uiuc.edu/.

[28] Helena R Lourenço, Olivier C Martin, and Thomas Stutzle. Iterated local search. *arXiv preprint math/0102188*, 2001.

[29] Vladimir J Lumelsky and Alexander A Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430, 1987.

[30] Theodore W Manikas, Kaveh Ashenayi, and Roger L Wainwright. Genetic algorithms for autonomous robot navigation. *Instrumentation & Measurement Magazine, IEEE*, 10(6):26–31, 2007.

[31] Marco Pavone, Beçhet Açikmeşe, Issa Nesnas, and Joseph A. Starek. Spacecraft Autonomy Challenges for Next Generation Space Missions. In *Springer Lecture Notes in Control and Information Sciences*. 2014. Submitted.

[32] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian P. Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *ICRA*, pages 300–307. IEEE, 2010.

[33] Ellips Masehian and Mohammad Reza Amin-Naseri. Sensor-based robot motion planning-a tabu search approach. *Robotics & Automation Magazine, IEEE*, 15(2):48–57, 2008.

[34] Luis A. Mateos and Markus Vincze. In-pipe robot with capability of self stabilization and accurate pipe surface cleaning. In *Proceedings of the IEEE International Conference on Automation Science and Engineering 2013*, 2013.

[35] R. McAllister, T. Peynot, R. Fitch, and S. Sukkarieh. Motion planning and stochastic control with experimental validation on a planetary rover. In

Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 4716–4723, Oct 2012. doi: 10.1109/IROS.2012.6386229.

[36] Hao Mei, Yantao Tian, and Linan Zu. A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. *International Journal of Information Technology*, 12(3):78–88, 2006.

[37] Javier Minguez and Luis Montano. Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. *Robotics and Automation, IEEE Transactions on*, 20(1):45–59, 2004.

[38] Javier Minguez, Luis Montano, Thierry Siméon, and Rachid Alami. Global nearness diagram navigation (GND). In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 33–39. IEEE, 2001.

[39] Javier Minguez, Javier Osuna, and Luis Montano. A" divide and conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3855–3862. IEEE, 2004.

[40] Petter Ogren and Naomi Ehrich Leonard. A convergent dynamic window approach to obstacle avoidance. *Robotics, IEEE Transactions on*, 21(2):188–195, 2005.

[41] Min Gyu Park, Jae Hyun Jeon, and Min Cheol Lee. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 3, pages 1530–1535. IEEE, 2001.

[42] Roland Philippsen. *Motion Planning and Obstacle Avoidance for Mobile Robots in Highly Cluttered Dynamic Environments*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2004. Thèse sciences EPFL no 3146 (2004), Faculté des sciences et techniques de l'ingénieur STI, Section de microtechnique, Institut d'ingénierie des systèmes.

[43] Maria Isabel Ribeiro. Obstacle avoidance. 2005. URL http://users.isr.ist.utl.pt/~mir/pub/ObstacleAvoidance.pdf.

[44] C. Schuetz, J. Baur, J. Pfaff, T. Buschmann, and H. Ulbrich. Multipurpose redundant manipulators for agricultural tasks. In *Proceedings of the Austrian Robotics Workshop 2014, Linz, Austria*, May 2014.

[45] Marija Seder and Ivan Petrovic. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *ICRA*, pages 1986–1991. IEEE, 2007. URL http://dblp.uni-trier.de/db/conf/icra/icra2007.html#SederP07.

[46] Reid Simmons. The curvature-velocity method for local obstacle avoidance. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3375–3382. IEEE, 1996.

[47] M. Suchi, M. Bader, and M. Vincze. Meta-heuristic search strategies for local path-planning to find collision free trajectories. In *Proceedings of the Austrian Robotics Workshop 2014, Linz, Austria*, May 2014.

[48] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. accepted for publication.

[49] Edward P. K. Tsang and Christos Voudouris. Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Oper. Res. Lett.*, 20(3):119–127, 1997.

[50] Iwan Ulrich and Johann Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1572–1577. IEEE, 1998.

[51] Iwan Ulrich and Johann Borenstein. Vfh^*: Local obstacle avoidance with look-ahead verification. In *ICRA*, pages 2505–2511, 2000.

[52] Christos Voudouris and Edward P. K. Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499, 1999.

[53] David H Wolpert and William G Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[54] W.L. Zagler, P. Mayer, P. Panek, M. Vincze, A. Weiss, M. Bajones, P. Puente, A. Huber, L. Lammer, and D. Fischinger. Roboter-Unterstützung zu Hause - Das Projekt HOBBIT. In *Proceedings of the 7th Deutscher AAL-Kongress, Berlin, ISBN: 978-3-8007-3574-7.*, 2014.

[55] Kunli Zhou, Song Ma, Xuliang Zhu, Lei Tang, and Xinhuan Feng. Improved ant colony algorithm based on cellular automata for obstacle avoidance in robot soccer. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 5, pages 298–302. IEEE, 2010.